TECHNICAL REFERENCE MANUAL

NVIDIA TEGRA 2 Family

NVIDIA® Tegra® 2 Series Mobile Processors: Tegra 250, Tegra 250S, AP20H

Abstract

All information contained in this document may not be applicable to all parts listed. Information that does not apply to all parts will indicate the part(s) that it applies to; for example, Tegra 250 Only or AP20 Only.

The Technical Reference Manual focuses on the logical organization and control of Tegra 2 Series devices. It provides information for those modules that interface to external devices, or those that control fundamental chip operations. The modules detailed in this document provide an overview, any necessary programming guidelines, and a register listing for that module. Internal functional units such as video and graphics hardware acceleration are controlled by NVIDIA provided software and not documented here.

Revision History

Version	Date	Description
v01p	SEP 14, 2011	Public release to support open source development

Table of Contents

1.0 INTRODUCTION

NVIDIA® Tegra® 2 Series devices are complete applications and digital media systems built around these processing elements:

- CPU Complex: Dual Core ARM® Cortex-A9 Symmetric MPCore™ processor. This is ARM's most advanced processor core: with dual instruction issue and both out-of-order and speculative execution. It has full cache coherency support for the dual symmetric processors. Both processors have 32 KB Instruction and 32 KB Data Level 1 caches; and there is a 1 MB shared Level 2 cache.
- Audio/Video Decoder: the AVP (Audio-Video Processor) subsystem includes dedicated audio and video decode hardware acceleration, an ARM7 processor, and embedded RAM. This subsystem provides full motion playback of up to 1080P high-definition video and supports the H.264, VC-1, and MPEG-4 video standards and multiple audio standards with dedicated hardware.
- Video Encoder: A high performance H.264 1080P capable hardware video encoder.
  - H.264 Baseline Profile Encoding:
    - Motion estimation using all partition types (16x16, 16x8, 8x16, 8x8, 4x8, 8x4, 4x4)
    - Motion estimation at integer and sub-pel (half pel and quarter pel) resolutions
    - Motion estimation using one reference frame
    - CBR and VBR rate control
    - Periodic intra frame insertion
    - Intra mode decision using all 4 intra16x16 sub modes
    - Intra mode decision using all 9 intra 4x4 sub modes
    - Recon loop, de-blocking, CAVLC conforming to H.264 standard
    - Quantization post processing
    - IPCM macro block support
    - Programmable intra refresh for error resiliency
    - FMO (up to 3 slice groups)
    - Macro block based and bit based packetization (multiple slice)
  - MPEG4 Simple Profile Encoding:
    - Motion estimation using 16x16 and 8x8 partitions
    - Motion estimation at full pel and half pel resolution
    - CBR rate control
    - AC DC prediction
    - Data partition
    - VLC conforming to MPEG4 standard
    - Short video header mode (H.263 Profile 0)
    - Macro block based and bit based packetization (multiple slice)
- Graphics: Ultra Low-power NVIDIA® GeForce® Graphics Processing Unit (GPU). Handles 2D graphics rendering and 3D pixel and vertex shading.
- Imaging: A high quality hardware accelerated still-image and video capture path supporting Bayer and YUV format sensors.
- Display: Dual display controllers with various output options for LCD panels and televisions, including HDMI output at up to 1080p.

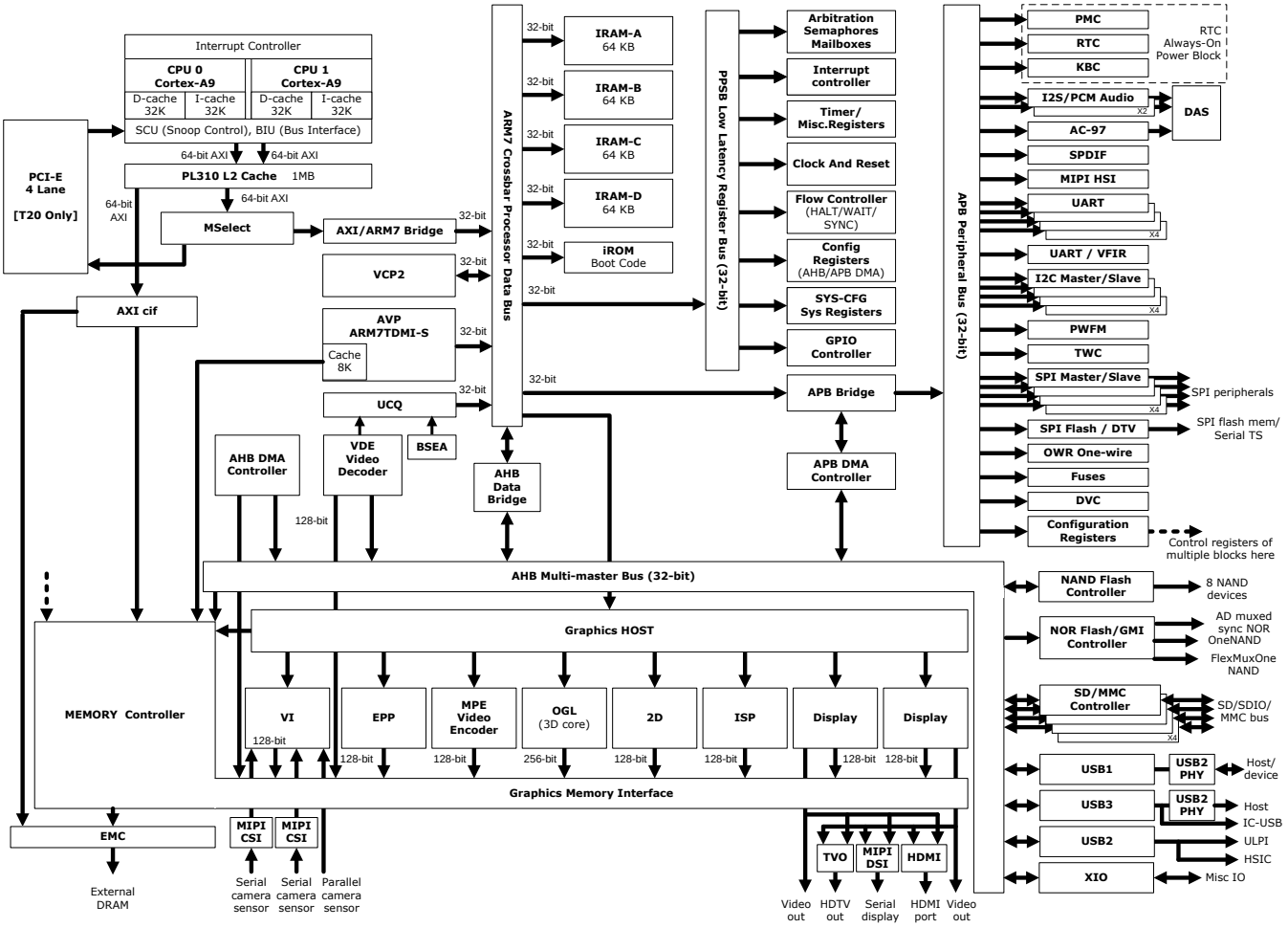
In addition to its processing elements, Tegra 2 Series has a broad range of peripheral interfaces to enable communication with wireless baseband, other communications peripherals, audio codecs, power management, and mass storage. When combined with baseband and PMU chips, the Tegra 2 Series devices provide the functionality needed to build a smart-phone. Dedicated high-performance mass storage controllers, with their own DMA engines, free the CPU Complex from routine data management tasks.

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

1.1 Block Diagram

This diagram provides an overview of a Tegra 2 Series Processor.

Figure 1. Tegra 2 Series Block Diagram



1.2 Memory Controller and Internal Bus Architecture

Tegra 2 Series Processor has a highly optimized memory controller, supporting low latency access for the CPU, optimized high bandwidth access for the graphics and video devices, and controlled latency for real time devices such as display.

There is a three level hierarchy of memory clients:

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

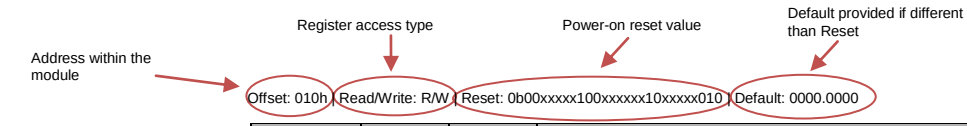
Special provision is made for the CPU to bypass much of the memory controller arbitration. This is the low latency path.

1.3 Reading Register Tables

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

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

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



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

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

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

Reset:
0bxxxxxxx00xxxxx100xxxxxx10xxxxx010
vs.
0b00xxxxx100xxxxxx10xxxxx010

Bit	R/W	Reset	Description
25:24	RW	N1	EMEM_NUMDEV 0 = N1 1 = N2
18:16	RO	D64MB	EMEM_DEVSIZE 0 = D4 MB 1 = D8 MB 2 = D16 MB 3 = D32 MB 4 = D64 MB 5 = D128 MB 6 = D256 MB 7 = D512 MB
9:8	RW	W2	EMEM_BANKWIDTH 1 = W1 2 = W2 3 = W3
2:0	RO	W9	EMEM_COLWIDTH 0 = W7 1 = W8 2 = W9 3 = W10 4 = W11

1.4 Glossary

Term	Definition
AHB	AMBA High-Speed Bus, a multi-master high-speed (relative to APB) bus supporting arbitration and split transactions, defined as part of AMBA 2.
AMBA™	Advanced Microcontroller Bus Architecture, a set of standard buses defined by ARM®.
APB	AMBA Peripheral Bus, a simple 32-bit single master bus for peripheral devices.
AVP	Audio-Video Processor, the term used both to describe the ARM7 processor in Tegra 2 Series devices, and to describe the broader audio and video decode acceleration hardware and RAM associated with the ARM7. Note that the AVP is sometimes known as COP in legacy documentation and registers.
AXI	AMBA Advanced eXtensible Interface, a more advanced bus than AHB defined as part of AMBA 3.
BSEA	Bit Stream Engine for Audio applications
CAR	Clock and Reset module allows controlling clocks and resets to all the modules and subsystems in Tegra 2 Series devices.
COP	CO-Processor, an obsolete name for the AVP still present in legacy documentation and registers.
CSI	MIPI Camera Serial Interface, a standard high-speed serial interface for connecting cameras to Tegra 2 Series devices.
CVE	Zoran TV Encoder
DSI	MIPI Display Serial Interface, a standard high-speed serial interface for connecting displays to Tegra 2 Series devices.
DVC	Dynamic Voltage Controller module
EMC	External Memory Controller, a module that interfaces with external DDR/LPDDR devices.
EPP	The video Encoder Pre-Processor, on Tegra 2 Series devices capable of filtering, color-space conversion, rotation and pixel format conversion.
HDMI	High-Definition Multimedia Interface, a digital connection carrying uncompressed video and audio at high speed over a single connector.
HSI	MIPI High-Speed Synchronous Interface, a standard high-speed serial interface for bi-directional communications with baseband processors and other devices.
IDE	Integrated Drive Electronics (or Integrated Device Electronics)
ISP	Image Signal Processor
KBC	Keyboard Controller module allows Tegra 2 Series devices to be connected to keyboard matrices of sizes up to 16x8.
MC	Memory Controller module handles requests from internal clients and arbitrates among them to allocate memory bandwidth.
MIPI	The Mobile Industry Processor Interface and industry alliance promoting a number of standard interfaces for mobile devices.
MPE	Video Encoder in Tegra 2 Series devices capable of encoding raw video stream into MPEG
NAND	A type of flash memory supporting high densities, and commonly used for non-volatile mass storage in portable devices. Accessed in sequential blocks to be generally treated as a file system.
NOR	A type of flash memory, with a direct bus interface that allows random access so code can be executed in place. Generally more costly and less dense than NAND flash memory.
OGL	Open Graphics Library, also known as OpenGL. An API supported on Tegra 2 Series devices and accelerated in hardware by dedicated 3D and 2D engines.
PMC	Power Management Controller module controls the various power management features in the system.
PWFM	Pulse Width Frequency Modulation module generates programmed pulse widths typically used to control backlight in display panels.

Term	Definition
PVT	Process, Voltage & Temperature
SMP	Symmetric Multi-Processing
SOC	System On a Chip, an integrated circuit containing a CPU, memory controller and the peripheral devices needed for a computing system.
S/PDIF	Sony/Philips Digital Interconnect Format
TWC	Three-Wire Controller
TVO	TV Encoder Output
UCQ	Unified Command Queue – a sub module within Video Decoder Engine
VCP2	Vector Co-Processor version 2, a hardware acceleration block for the signal processing parts of audio decode and filtering. Use to offload the ARM7 AVP during audio playback.
VDE	Video Decode Engine, the Tegra 2 Series hardware acceleration block dedicated to decoding compressed video in various formats.
VI	Video Input block, the acronym used to describe the Tegra 2 Series block used for camera and related input functions.

2.0 ADDRESS AND INTERRUPT MAP

2.1 System Memory Map

Table 1. System Memory Map

Description	Address Start	Address End	Default Length
External DRAM Memory (EMEM)	0000:0000	3FFF:FFFF	1 GB
Embedded AVP memory (IRAM)	4000:0000	4003:FFFF	256 KB
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
ARM Cortex™-A9 CPU registers	5004:0000	5004:1FFF	8 KB
ARM PERIPHBASE	5004:0000	5004:1FFF	8 KB
ARM Interrupt distributor	5004:1000	5004:1FFF	4 KB
MSelect Register Space	5004:2000	5004:2FFF	4 KB
PL310 (L2 Cache controller)	5004:3000	5004:3FFF	4 KB
Graphics Host registers	5400:0000	547F:FFFF	64 MB
MPE	5404:0000	5407:FFFF	256 KB
VI and CSI	5408:0000	540B:FFFF	256 KB
EPP	540C:0000	540F:FFFF	256 KB
ISP	5410:0000	5413:FFFF	256 KB
GR2D	5414:0000	5417:FFFF	256 KB
GR3D	5418:0000	541B:FFFF	256 KB
DISPLAY	5420:0000	5423:FFFF	256 KB
DISPLAY	5424:0000	5427:FFFF	256 KB
HDMI	5428:0000	542B:FFFF	256 KB
TVO	542C:0000	542F:FFFF	256 KB
DSI	5430:0000	5433:FFFF	256 KB
GART	5800:0000	59FF:FFFF	32 MB
PPSB Bus device registers	6000:0000	6FFF:FFFF	256 MB
uP-TAG	6000:0000	6000:0FFF	4 KB
Resource Semaphore	6000:1000	6000:1FFF	4 KB
Arbitration Semaphore	6000:2000	6000:2FFF	4 KB
ARB-PRI	6000:3000	6000:3FFF	4 KB
Primary ICTLR	6000:4000	6000:403F	64 B
Primary ICTLR ARB-GNT	6000:4040	6000:40FF	192 B
Secondary ICTLR	6000:4100	6000:413F	64 B

Description	Address Start	Address End	Default Length
Secondary ICTLR DMA TX	6000:4140	6000:4147	8 B
Secondary ICTLR DMA RX	6000:4148	6000:414F	8 B
Tertiary ICTLR	6000:4200	6000:423F	64 B
Quaternary ICTLR	6000:4300	6000:433F	64 B
TMR1	6000:5000	6000:5007	8 B
TMR2	6000:5008	6000:500F	8 B
TMRUS	6000:5010	6000:504F	64 B
TMR3	6000:5050	6000:5057	8 B
TMR4	6000:5058	6000:505F	8 B
CLK and RESET	6000:6000	6000:6FFF	4 KB
Flow Controller	6000:7000	6000:7013	20 B
AHB-DMA	6000:8000	6000:8FFF	4 KB
AHB-DMA CH0	6000:9000	6000:901F	32 B
AHB-DMA CH1	6000:9020	6000:903F	32 B
AHB-DMA CH2	6000:9040	6000:905F	32 B
AHB-DMA CH3	6000:9060	6000:907F	32 B
APB-DMA	6000:A000	6000:AFFF	4 KB
APB-DMA CH0	6000:B000	6000:B01F	32 B
APB-DMA CH1	6000:B020	6000:B03F	32 B
APB-DMA CH2	6000:B040	6000:B05F	32 B
APB-DMA CH3	6000:B060	6000:B07F	32 B
APB-DMA CH4	6000:B080	6000:B09F	32 B
APB-DMA CH5	6000:B0A0	6000:B0BF	32 B
APB-DMA CH6	6000:B0C0	6000:B0DF	32 B
APB-DMA CH7	6000:B0E0	6000:B0FF	32 B
APB-DMA CH8	6000:B100	6000:B11F	32 B
APB-DMA CH9	6000:B120	6000:B13F	32 B
APB-DMA CH10	6000:B140	6000:B15F	32 B
APB-DMA CH11	6000:B160	6000:B17F	32 B
APB-DMA CH12	6000:B180	6000:B19F	32 B
APB-DMA CH13	6000:B1A0	6000:B1BF	32 B
APB-DMA CH14	6000:B1C0	6000:B1DF	32 B
APB-DMA CH15	6000:B1E0	6000:B1FF	32 B
CACHE (ARM7 Cache Controller)	6000:C000	6000:C3FF	1 KB
AHB Arbitration + Bus Interfaces	6000:C004	6000:C10F	268 B
AHB/APB Debug Bus	6000:C150	6000:C1F5	166 B
Secure Boot	6000:C200	6000:C203	4 B
STAT-MON	6000:C400	6000:C7FF	1 KB

Description	Address Start	Address End	Default Length
GPIO-1	6000:D000	6000:D87F	2176 B
GPIO-2	6000:D080	6000:D8FF	2176 B
GPIO-3	6000:D100	6000:D97F	2176 B
GPIO-4	6000:D180	6000:D9FF	2176 B
GPIO-5	6000:D200	6000:DA7F	2176 B
GPIO-6	6000:D280	6000:DAFF	2176 B
GPIO-7	6000:D300	6000:DB7F	2176 B
VCP	6000:E000	6000:EFFF	4 KB
Exception vectors	6000:F000	6000:FFFF	4 KB
UCQ	6001:0000	6001:00FF	256 B
BSEA	6001:1000	6001:1FFF	4 KB
SXE	6001:A000	6001:AFFF	4 KB
BSEV	6001:B000	6001:BFFF	4 KB
MBE	6001:C000	6001:C0FF	256 B
PPE	6001:C200	6001:C2FF	256 B
MCE	6001:C400	6001:C4FF	256 B
TFE	6001:C600	6001:C6FF	256 B
PPB	6001:C800	6001:C8FF	256 B
VDMA	6001:CA00	6001:CAFF	256 B
UCQ	6001:CC00	6001:CCFF	256 B
BSEA	6001:D000	6001:D7FF	2 KB
FRAMEID	6001:D800	6001:DAFF	768 B
APB Bus Registers	7000:0000	7FFF:FFFF	256 MB
MISC	7000:0000	7000:0FFF	4 KB
AC97	7000:2000	7000:21FF	512 B
SPDIF	7000:2400	7000:25FF	512 B
I2S Audio 1	7000:2800	7000:28FF	256 B
I2S Audio 2	7000:2A00	7000:2AFF	256 B
UARTA (UART1)	7000:6000	7000:603F	64 B
UARTB (UART2)	7000:6040	7000:607F	64 B
VFIR	7000:6100	7000:61FF	256 B
UARTC (UART3)	7000:6200	7000:62FF	256 B
UARTD (UART4)	7000:6300	7000:63FF	256 B
UARTE (UART5)	7000:6400	7000:64FF	256 B
NAND Controller	7000:8000	7000:80FF	256 B
S-NOR Controller	7000:9000	7000:9FFF	4 KB
PWFM Controller	7000:A000	7000:A0FF	256 B
MIPI-HSI Baseband	7000:B000	7000:B0FF	256 B

Description	Address Start	Address End	Default Length
I2C	7000:C000	7000:C0FF	256 B
TWC	7000:C100	7000:C1FF	256 B
SPI	7000:C380	7000:C3AF	48 B
I2C2	7000:C400	7000:C4FF	256 B
I2C3	7000:C500	7000:C5FF	256 B
OWR	7000:C600	7000:C64F	80 B
DVC	7000:D000	7000:D1FF	512 B
SPI 1 (SLINK 2B)	7000:D400	7000:D5FF	512 B
SPI 2 (SLINK 2B)	7000:D600	7000:D7FF	512 B
SPI 3 (SLINK 2B)	7000:D800	7000:D9FF	512 B
SPI 4 (SLINK 2B)	7000:DA00	7000:DBFF	512 B
RTC	7000:E000	7000:E0FF	256 B
KBC	7000:E200	7000:E2FF	256 B
PMC	7000:E400	7000:E4FF	256 B
MC	7000:F000	7000:F3FF	1 KB
EMC	7000:F400	7000:F7FF	1 KB
FUSE	7000:F800	7000:FBFF	1 KB
KFUSE	7000:FC00	7000:FFFF	1 KB
CoreSight™ Debug Registers (CSITE)	7004:0000	7007:FFFF	256 KB
AHB External Mem (AVP/AHB only)	8000:0000	BFFF:FFFF	1 GB
PCIE Bus (CPU only) – (Tegra 250 ONLY)	8000:0000	BFFF:FFFF	1 GB
AHB Bus	C000:0000	CFFF:FFFF	256 MB
EIDE **– (Tegra 250 ONLY)**	C300:0000	C3FF:FFFF	16 MB
PPCS (AHB to MC flush)	C400:0000	C400:FFFF	64 KB
USB	C500:0000	C500:3FFF	16 KB
USB2	C500:4000	C500:7FFF	16 KB
USB3	C500:8000	C500:BFFF	16 KB
SDMMC-1	C800:0000	C800:01FF	512 B
SDMMC-2	C800:0200	C800:03FF	512 B
SDMMC-3	C800:0400	C800:05FF	512 B
SDMMC-4	C800:0600	C800:07FF	512 B
NOR Flash	D000:0000	DFFF:FFFF	256 MB
MMU TLB	F000:F000	F000:FFFF	4 KB
IROM (boot code)	FFF0:0000	FFF0:BFFF	48 KB
Hi-VEC	FFFF:0000	FFFF:FFFF	64 KB

2.2 Access Restrictions

Not every memory region can be accessed by every device. The next table shows the restrictions by class of master on each address region.

Table 2. Address Map Restrictions by Master

Address range	CPU	AVP	AHB masters / APB DMA	MCCIF clients
0000 0000 - 0000 003F	DRAM	ivector	DRAM	DRAM
0000 0040 - 3FFF FFFF	DRAM	DRAM (cached)	DRAM	DRAM
4000 0000 - 4003 FFFF	IRAM	IRAM	IRAM	not accessible
4004 0000 - 4FFF FFFF	unused	unused	unused	unused
5000 0000 - 5002 3FFF	Host registers	Host registers	Host registers	not accessible
5002 4000 - 5003 FFFF	unused	unused	unused	unused
5004 0000 - 5004 1FFF	ARM registers	not accessible	not accessible	not accessible
5004 2000 - 5004 2FFF	Mselect registers	not accessible	not accessible	not accessible
5004 3000 - 5004 3FFF	PL310 (L2 cache)	not accessible	not accessible	not accessible
5004 3000 - 52FF FFFF	unused	unused	unused	unused
5300 0000 - 53FF FFFF	unused (verification aperture)	unused	unused	unused
5400 0000 - 57FF FFFF	Direct access to host clients	Direct access to host clients	Direct access to host clients	not accessible
5800 0000 - 59FF FFFF	GART (no coherency)	GART (non-cached)	GART	GART
5A00 0000 - 5FFF FFFF	unused	unused	unused	unused
6000 0000 - 6FFF FFFF	PPSB	PPSB	PPSB	not accessible
7000 0000 - 7FFF FFFF	APB	APB	APB	not accessible
8000 0000 - BFFF FFFF	PCIE **– (Tegra 250 ONLY)**	DRAM (non cached)	DRAM	not accessible
C000 0000 - C3FF FFFF	AHB	AHB	AHB	not accessible
C400 0000 - C400 FFFF	Flush AHB DRAM access	Flush AHB DRAM access	Flush AHB DRAM access	not accessible
C401 0000 - C800 05FF	AHB	AHB	AHB	not accessible
C800 0600 - CFFF FFFF	unused	unused	unused	unused
D000 0000 - DFFF FFFF	NOR flash	NOR flash	NOR flash	unused
E000 0000 - EFFF FFFF	MIO	MIO	MIO	unused
F000 0000 - F000 FFFF	CCH (COP MMU)	CCH (COP MMU)	not accessible	not accessible
F001 0000 - FFFE FFFF	unused	unused	unused	unused
FFF0 0000 - FFF0 FFFF	IROM	IROM	IROM	not accessible
FFFF 0000 - FFFF FFFF	Hi-VEC	Hi-VEC	not accessible	not accessible

2.3 Interrupt Mapping

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

Interrupts to the Cortex-A9 embedded interrupt controller (GIC) are in this same order, but start at offset 32 as the first 32 are reserved for the CPU's internal interrupts. See the Interrupt Controller section of this document for more information on how interrupts are routed.

Table 3. Primary Interrupt Controller Mapping

Block Name	Interrupt Controller	Interrupt Number
TMR1	PRI_ICTLR	0
TMR2	PRI_ICTLR	1
RTC	PRI_ICTLR	2
I2S2	PRI_ICTLR	3
SHR-SEM Inbox IBF	PRI_ICTLR	4
SHR-SEM Inbox IBE	PRI_ICTLR	5
SHR-SEM Outbox IBF	PRI_ICTLR	6
SHR-SEM Outbox IBE	PRI_ICTLR	7
VDE UCQ Error Interrupt	PRI_ICTLR	8
VDE Sync Token Interrupt	PRI_ICTLR	9
VDE BSE-V Interrupt	PRI_ICTLR	10
VDE BSE-A Interrupt	PRI_ICTLR	11
VDE SXE Interrupt	PRI_ICTLR	12
I2S1	PRI_ICTLR	13
SDMMC1	PRI_ICTLR	14
SDMMC2	PRI_ICTLR	15
XIO	PRI_ICTLR	16
VDE Interrupt	PRI_ICTLR	17
AVP_UCQ	PRI_ICTLR	18
SDMMC3	PRI_ICTLR	19
USB	PRI_ICTLR	20
USB2	PRI_ICTLR	21
<unused>	PRI_ICTLR	22
EIDE **– (Tegra 250 ONLY)**	PRI_ICTLR	23
NANDFLASH	PRI_ICTLR	24
VCP	PRI_ICTLR	25
APB_DMA	PRI_ICTLR	26
AHB_DMA	PRI_ICTLR	27
GNT.0 Arbitration Grant Status of COP	PRI_ICTLR	28
GNT.1 Arbitration Grant Status of CPU	PRI_ICTLR	29
OWR	PRI_ICTLR	30

Block Name	Interrupt Controller	Interrupt Number
SDMMC4	PRI_ICTLR	31

Table 4. Secondary Interrupt Controller Mapping

Block Name	Interrupt Controller	Interrupt Number
GPIO1	SEC_ICTLR	0
GPIO2	SEC_ICTLR	1
GPIO3	SEC_ICTLR	2
GPIO4	SEC_ICTLR	3
UARTA (UART1)	SEC_ICTLR	4
UARTB (UART2)	SEC_ICTLR	5
I2C	SEC_ICTLR	6
SPI - SFLASH	SEC_ICTLR	7
TWC	SEC_ICTLR	8
TMR3	SEC_ICTLR	9
TMR4	SEC_ICTLR	10
FLOW - RSM0 Resume for CPU	SEC_ICTLR	11
FLOW - RSM1 Resume for COP	SEC_ICTLR	12
SPDIF	SEC_ICTLR	13
UARTC (UART3)	SEC_ICTLR	14
MIPI-HSI baseband controller	SEC_ICTLR	15
EVENTA	SEC_ICTLR	16
EVENTB	SEC_ICTLR	17
EVENTC	SEC_ICTLR	18
EVENTD	SEC_ICTLR	19
VFIR	SEC_ICTLR	20
DVC	SEC_ICTLR	21
SYS_STATS_MON	SEC_ICTLR	22
GPIO5	SEC_ICTLR	23
CPU0_PMU_INTR	SEC_ICTLR	24
CPU1_PMU_INTR	SEC_ICTLR	25
<unused>	SEC_ICTLR	26
S-LINK1 (SPI1)	SEC_ICTLR	27
APB_DMA_COP	SEC_ICTLR	28
AHB_DMA_COP	SEC_ICTLR	29
DMA TX Interrupt	SEC_ICTLR	30
DMA RX Interrupt	SEC_ICTLR	31

Table 5. Tertiary Interrupt Controller Mapping

Block Name	Interrupt Controller	Interrupt Number
HOST1X - COP SyncPt Interrupt	TRI_ICTLR	0
HOST1X - CPU SyncPt Interrupt	TRI_ICTLR	1
HOST1X - COP General Interrupt	TRI_ICTLR	2
HOST1X - CPU General Interrupt	TRI_ICTLR	3
MPE General Interrupt	TRI_ICTLR	4
VI General Interrupt	TRI_ICTLR	5
EPP General Interrupt	TRI_ICTLR	6
ISP General Interrupt	TRI_ICTLR	7
2D General Interrupt	TRI_ICTLR	8
Display General Interrupt	TRI_ICTLR	9
Display B General Interrupt	TRI_ICTLR	10
HDMI INT from pins	TRI_ICTLR	11
TVO General Interrupt	TRI_ICTLR	12
MC General Interrupt	TRI_ICTLR	13
EMC General Interrupt	TRI_ICTLR	14
<unused>	TRI_ICTLR	15
<unused>	TRI_ICTLR	16
AC97	TRI_ICTLR	17
SPI 2 (SBC2)	TRI_ICTLR	18
SPI 3 (SBC3)	TRI_ICTLR	19
I2C2	TRI_ICTLR	20
KBC	TRI_ICTLR	21
External PMU	TRI_ICTLR	22
GPIO6	TRI_ICTLR	23
TVDAC (from MISC)	TRI_ICTLR	24
GPIO7	TRI_ICTLR	25
UARTD (UART4)	TRI_ICTLR	26
UARTE (UART5)	TRI_ICTLR	27
I2C3	TRI_ICTLR	28
SPI 4 (SBC4)	TRI_ICTLR	29
<unused>	TRI_ICTLR	30
SW Reserved Interrupt	TRI_ICTLR	31

Table 6. Quaternary Interrupt Controller Mapping

Block Name	Interrupt Controller	Interrupt Number
SNOR	QUAD_ICTLR	0
USB3	QUAD_ICTLR	1
PCIE_**INTR– (Tegra 250 ONLY)**	QUAD_ICTLR	2
PCIE_MSI **– (Tegra 250 ONLY)**	QUAD_ICTLR	3
<unused>	QUAD_ICTLR	4
<unused>	QUAD_ICTLR	5
<unused>	QUAD_ICTLR	6
<unused>	QUAD_ICTLR	7
APB_DMA_CH0	QUAD_ICTLR	8
APB_DMA_CH1	QUAD_ICTLR	9
APB_DMA_CH2	QUAD_ICTLR	10
APB_DMA_CH3	QUAD_ICTLR	11
APB_DMA_CH4	QUAD_ICTLR	12
APB_DMA_CH5	QUAD_ICTLR	13
APB_DMA_CH6	QUAD_ICTLR	14
APB_DMA_CH7	QUAD_ICTLR	15
APB_DMA_CH8	QUAD_ICTLR	16
APB_DMA_CH9	QUAD_ICTLR	17
APB_DMA_CH10	QUAD_ICTLR	18
APB_DMA_CH11	QUAD_ICTLR	19
APB_DMA_CH12	QUAD_ICTLR	20
APB_DMA_CH13	QUAD_ICTLR	21
APB_DMA_CH14	QUAD_ICTLR	22
APB_DMA_CH15	QUAD_ICTLR	23
<unused>	QUAD_ICTLR	24
<unused>	QUAD_ICTLR	25
<unused>	QUAD_ICTLR	26
<unused>	QUAD_ICTLR	27
<unused>	QUAD_ICTLR	28
<unused>	QUAD_ICTLR	29
<unused>	QUAD_ICTLR	30
<unused>	QUAD_ICTLR	31

2.4 APB DMA Interrupts

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

Table 7. APB DMA TX Interrupt Mapping

Block Name	Interrupt Controller	Interrupt Number
AC97_PB	DMA_TX_INTR	0
I2S1	DMA_TX_INTR	1
I2S1	DMA_TX_INTR	2
SPDIF	DMA_TX_INTR	3
SPDIF	DMA_TX_INTR	4
I2S2	DMA_TX_INTR	5
I2S2	DMA_TX_INTR	6
SPI - SFLASH	DMA_TX_INTR	7
UARTA (UART1)	DMA_TX_INTR	8
UARTB (UART2)	DMA_TX_INTR	9
UARTC (UART3)	DMA_TX_INTR	10
TWC	DMA_TX_INTR	11
I2C	DMA_TX_INTR	12
I2C2	DMA_TX_INTR	13
<unused>	DMA_TX_INTR	14
<unused>	DMA_TX_INTR	15
<unused>	DMA_TX_INTR	16
VFIR	DMA_TX_INTR	17
AC97_MPB	DMA_TX_INTR	18
<unused>	DMA_TX_INTR	19
<unused>	DMA_TX_INTR	20
<unused>	DMA_TX_INTR	21
<unused>	DMA_TX_INTR	22
<unused>	DMA_TX_INTR	23
<unused>	DMA_TX_INTR	24
<unused>	DMA_TX_INTR	25
<unused>	DMA_TX_INTR	26
<unused>	DMA_TX_INTR	27
<unused>	DMA_TX_INTR	28
<unused>	DMA_TX_INTR	29
<unused>	DMA_TX_INTR	30
<unused>	DMA_TX_INTR	31

Table 8. APB DMA RX Interrupt Mapping

Block Name	Interrupt Controller	Interrupt Number
AC97_REC	DMA_RX_INTR	0
I2S1	DMA_RX_INTR	1
I2S1	DMA_RX_INTR	2
SPDIF	DMA_RX_INTR	3
SPDIF	DMA_RX_INTR	4
I2S2	DMA_RX_INTR	5
I2S2	DMA_RX_INTR	6
SPI - SFLASH	DMA_RX_INTR	7
UARTA (UART1)	DMA_RX_INTR	8
UARTB (UART2)	DMA_RX_INTR	9
UARTC (UART3)	DMA_RX_INTR	10
TWC	DMA_RX_INTR	11
I2C RX Slave	DMA_RX_INTR	12
I2C2	DMA_RX_INTR	13
<unused>	DMA_RX_INTR	14
<unused>	DMA_RX_INTR	15
<unused>	DMA_RX_INTR	16
VFIR	DMA_RX_INTR	17
AC97_MREC	DMA_RX_INTR	18
<unused>	DMA_RX_INTR	19
<unused>	DMA_RX_INTR	20
<unused>	DMA_RX_INTR	21
<unused>	DMA_RX_INTR	22
<unused>	DMA_RX_INTR	23
<unused>	DMA_RX_INTR	24
<unused>	DMA_RX_INTR	25
<unused>	DMA_RX_INTR	26
<unused>	DMA_RX_INTR	27
<unused>	DMA_RX_INTR	28
<unused>	DMA_RX_INTR	29
<unused>	DMA_RX_INTR	30
<unused>	DMA_RX_INTR	31

3.0 INTERRUPT CONTROLLER

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

The Tegra® 2 Series has four sets of Interrupt Controllers, each handling 32 possible interrupt sources. A flexible interrupt controller allows priority scheduling and Interrupt Request steering. This allows any interrupt request to be re-routed to either processor to allow dynamic redistribution of workload, resulting in increased efficiency.

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

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

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

The Tegra 2 Series devices can also use software DMA to handle DMA requests. The DMA interrupt requests are grouped into read requests and write requests and presented at bits 31 and 30 of the ISR register. Individual requests can be viewed by reading the DSR register. The Arbitration Semaphore module can also generate interrupt requests to the interrupt controller when a processor achieves Arbitration Grant Status. When a 1 is set in the proper bit position of the Arbitration Semaphore Interrupt Source Enable register (GNT0_CPU.ENABLE), that particular request causes an interrupt when the Grant Status becomes true.

3.1 Functionality

The programmable Interrupt Controller supports up to 128 interrupts and is built from four 32-bit interrupt controllers - Primary, Secondary, Tertiary and Quaternary.

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

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

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

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

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

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

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

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

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

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

3.1.1 Interrupt Function Mapping

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

3.2 Interrupt Registers

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

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

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

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

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

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

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

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

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

3.2.1 Primary Interrupt Controller Registers

3.2.1.1 PRI_ICTLR_VIRQ_CPU_0

Valid Interrupt Request Status for CPU Register

Offset: 000h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	SDMMC4
30	X	OWR
29	X	ARB_SEM_GNT_CPU
28	X	ARB_SEM_GNT_COP
27	X	AHB_DMA_CPU
26	X	APB_DMA_CPU
25	X	VCP
24	X	NANDCTRL
23	X	EIDE
22	X	SDMMC4
21	X	USB2
20	X	USB
19	X	SDMMC3
18	X	AVP_UCQ
17	X	VDE
16	X	XIO
15	X	SDMMC2
14	X	SDMMC1
13	X	I2S1
12	X	VDE_SXE
11	X	VDE_BSEA

Bit	Reset	Description
10	X	VDE_BSEV
9	X	VDE_SYNC_TOKEN
8	X	VDE_UCQ
7	X	SHR_SEM_OUTBOX_EMPTY
6	X	SHR_SEM_OUTBOX_FULL
5	X	SHR_SEM_INBOX_EMPTY
4	X	SHR_SEM_INBOX_FULL
3	X	I2S2
2	X	RTC
1	X	TMR2
31:0	X	IRQ31_IRQ0: Flags set by Hardware, cleared by SW
0	X	TMR1

3.2.1.2 PRI_ICTLR_VIRQ_COP_0

Valid Interrupt Status for AVP (COP) Register

Offset: 004h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

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

Bit	Reset	Description
16	X	XIO
15	X	SDMMC2
14	X	SDMMC1
13	X	I2S1
12	X	VDE_SXE
11	X	VDE_BSEA
10	X	VDE_BSEV
9	X	VDE_SYNC_TOKEN
8	X	VDE_UCQ
7	X	SHR_SEM_OUTBOX_EMPTY
6	X	SHR_SEM_OUTBOX_FULL
5	X	SHR_SEM_INBOX_EMPTY
4	X	SHR_SEM_INBOX_FULL
3	X	I2S2
2	X	RTC
1	X	TMR2
31:0	X	IRQ31_IRQ0: Flags set by Hardware, cleared by SW
0	X	TMR1

3.2.1.3 PRI_ICTLR_VFIQ_CPU_0 FIQ

Valid Interrupt Status for CPU Register

Offset: 008h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	SDMMC4
30	X	OWR
29	X	ARB_SEM_GNT_CPU
28	X	ARB_SEM_GNT_COP
27	X	AHB_DMA_CPU
26	X	APB_DMA_CPU
25	X	VCP
24	X	NANDCTRL
23	X	EIDE

Bit	Reset	Description
22	X	SDMMC4
21	X	USB2
20	X	USB
19	X	SDMMC3
18	X	AVP_UCQ
17	X	VDE
16	X	XIO
15	X	SDMMC2
14	X	SDMMC1
13	X	I2S1
12	X	VDE_SXE
11	X	VDE_BSEA
10	X	VDE_BSEV
9	X	VDE_SYNC_TOKEN
8	X	VDE_UCQ
7	X	SHR_SEM_OUTBOX_EMPTY
6	X	SHR_SEM_OUTBOX_FULL
5	X	SHR_SEM_INBOX_EMPTY
4	X	SHR_SEM_INBOX_FULL
3	X	I2S2
2	X	RTC
1	X	TMR2
31:0	X	FIQ31_FIQ0: Flags set by Hardware, cleared by SW
0	X	TMR1

3.2.1.4 PRI_ICTLR_VFIQ_COP_0

FIQ Valid Interrupt Status for AVP (COP) Register

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

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

Bit	Reset	Description
28	X	ARB_SEM_GNT_COP
27	X	AHB_DMA_CPU
26	X	APB_DMA_CPU
25	X	VCP
24	X	NANDCTRL
23	X	EIDE
22	X	SDMMC4
21	X	USB2
20	X	USB
19	X	SDMMC3
18	X	AVP_UCQ
17	X	VDE
16	X	XIO
15	X	SDMMC2
14	X	SDMMC1
13	X	I2S1
12	X	VDE_SXE
11	X	VDE_BSEA
10	X	VDE_BSEV
9	X	VDE_SYNC_TOKEN
8	X	VDE_UCQ
7	X	SHR_SEM_OUTBOX_EMPTY
6	X	SHR_SEM_OUTBOX_FULL
5	X	SHR_SEM_INBOX_EMPTY
4	X	SHR_SEM_INBOX_FULL
3	X	I2S2
2	X	RTC
1	X	TMR2
31:0	X	FIQ31_FIQ0: Flags set by Hardware, cleared by SW
0	X	TMR1

3.2.1.5 PRI_ICTLR_ISR_0

Latched Interrupt Status Register (Hardware)

Offset: 010h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	SDMMC4
30	X	OWR
29	X	ARB_SEM_GNT_CPU
28	X	ARB_SEM_GNT_COP
27	X	AHB_DMA_CPU
26	X	APB_DMA_CPU
25	X	VCP
24	X	NANDCTRL
23	X	EIDE
22	X	SDMMC4
21	X	USB2
20	X	USB
19	X	SDMMC3
18	X	AVP_UCQ
17	X	VDE
16	X	XIO
15	X	SDMMC2
14	X	SDMMC1
13	X	I2S1
12	X	VDE_SXE
11	X	VDE_BSEA
10	X	VDE_BSEV
9	X	VDE_SYNC_TOKEN
8	X	VDE_UCQ
7	X	SHR_SEM_OUTBOX_EMPTY
6	X	SHR_SEM_OUTBOX_FULL
5	X	SHR_SEM_INBOX_EMPTY
4	X	SHR_SEM_INBOX_FULL
3	X	I2S2

Bit	Reset	Description
2	X	RTC
1	X	TMR2
31:0	X	ISR31_ISR0: Read-only. Set by hardware event, cleared at source by software
0	X	TMR1

3.2.1.6 PRI_ICTLR_FIR_0

Forced Interrupt Status Register (Software)

Offset: 014h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	SDMMC4
30	X	OWR
29	X	ARB_SEM_GNT_CPU
28	X	ARB_SEM_GNT_COP
27	X	AHB_DMA_CPU
26	X	APB_DMA_CPU
25	X	VCP
24	X	NANDCTRL
23	X	EIDE
22	X	SDMMC4
21	X	USB2
20	X	USB
19	X	SDMMC3
18	X	AVP_UCQ
17	X	VDE
16	X	XIO
15	X	SDMMC2
14	X	SDMMC1
13	X	I2S1
12	X	VDE_SXE
11	X	VDE_BSEA
10	X	VDE_BSEV
9	X	VDE_SYNC_TOKEN

Bit	Reset	Description
8	X	VDE_UCQ
7	X	SHR_SEM_OUTBOX_EMPTY
6	X	SHR_SEM_OUTBOX_FULL
5	X	SHR_SEM_INBOX_EMPTY
4	X	SHR_SEM_INBOX_FULL
3	X	I2S2
2	X	RTC
1	X	TMR2
31:0	X	FIR31_FIR0: Read only: Set during write to FIR_SET, cleared during write to FIR_CLR
0	X	TMR1

3.2.1.7 PRI_ICTLR_FIR_SET_0

Force Interrupt Register Set

Offset: 018h | Read/Write: WO | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	SDMMC4
30	0x0	OWR
29	0x0	ARB_SEM_GNT_CPU
28	0x0	ARB_SEM_GNT_COP
27	0x0	AHB_DMA_CPU
26	0x0	APB_DMA_CPU
25	0x0	VCP
24	0x0	NANDCTRL
23	0x0	EIDE
22	0x0	SDMMC4
21	0x0	USB2
20	0x0	USB
19	0x0	SDMMC3
18	0x0	AVP_UCQ
17	0x0	VDE
16	0x0	XIO
15	0x0	SDMMC2

Bit	Reset	Description
14	0x0	SDMMC1
13	0x0	I2S1
12	0x0	VDE_SXE
11	0x0	VDE_BSEA
10	0x0	VDE_BSEV
9	0x0	VDE_SYNC_TOKEN
8	0x0	VDE_UCQ
7	0x0	SHR_SEM_OUTBOX_EMPTY
6	0x0	SHR_SEM_OUTBOX_FULL
5	0x0	SHR_SEM_INBOX_EMPTY
4	0x0	SHR_SEM_INBOX_FULL
3	0x0	I2S2
2	0x0	RTC
1	0x0	TMR2
31:0	0x0	FIR_SET: Set Forced Interrupt Bit. Writing a 1 will set an interrupt
0	0x0	TMR1

3.2.1.8 PRI_ICTLR_FIR_CLR_0

Force Interrupt Register Clear Register

Offset: 01ch | Read/Write: WO | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	SDMMC4
30	0x0	OWR
29	0x0	ARB_SEM_GNT_CPU
28	0x0	ARB_SEM_GNT_COP
27	0x0	AHB_DMA_CPU
26	0x0	APB_DMA_CPU
25	0x0	VCP
24	0x0	NANDCTRL
23	0x0	EIDE
22	0x0	SDMMC4
21	0x0	USB2

Bit	Reset	Description
20	0x0	USB
19	0x0	SDMMC3
18	0x0	AVP_UCQ
17	0x0	VDE
16	0x0	XIO
15	0x0	SDMMC2
14	0x0	SDMMC1
13	0x0	I2S1
12	0x0	VDE_SXE
11	0x0	VDE_BSEA
10	0x0	VDE_BSEV
9	0x0	VDE_SYNC_TOKEN
8	0x0	VDE_UCQ
7	0x0	SHR_SEM_OUTBOX_EMPTY
6	0x0	SHR_SEM_OUTBOX_FULL
5	0x0	SHR_SEM_INBOX_EMPTY
4	0x0	SHR_SEM_INBOX_FULL
3	0x0	I2S2
2	0x0	RTC
1	0x0	TMR2
31:0	0x0	FIR_CLR: Clear Forced Interrupt Bit: Writing a 1 will clear the forced interrupt
0	0x0	TMR1

3.2.1.9 PRI_ICTLR_CPU_IER_0

Enabled Interrupt Source for CPU Register

Offset: 020h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

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

Bit	Reset	Description
26	X	APB_DMA_CPU
25	X	VCP
24	X	NANDCTRL
23	X	EIDE
22	X	SDMMC4
21	X	USB2
20	X	USB
19	X	SDMMC3
18	X	AVP_UCQ
17	X	VDE
16	X	XIO
15	X	SDMMC2
14	X	SDMMC1
13	X	I2S1
12	X	VDE_SXE
11	X	VDE_BSEA
10	X	VDE_BSEV
9	X	VDE_SYNC_TOKEN
8	X	VDE_UCQ
7	X	SHR_SEM_OUTBOX_EMPTY
6	X	SHR_SEM_OUTBOX_FULL
5	X	SHR_SEM_INBOX_EMPTY
4	X	SHR_SEM_INBOX_FULL
3	X	I2S2
2	X	RTC
1	X	TMR2
31:0	X	IER31_IER0: Interrupt Enable Status. 0 = Disabled
0	X	TMR1

3.2.1.10 PRI_ICTLR_CPU_IER_SET_0

Set Interrupt Enable for CPU Register

Offset: 024h | Read/Write: WO | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	SDMMC4
30	0x0	OWR
29	0x0	ARB_SEM_GNT_CPU
28	0x0	ARB_SEM_GNT_COP
27	0x0	AHB_DMA_CPU
26	0x0	APB_DMA_CPU
25	0x0	VCP
24	0x0	NANDCTRL
23	0x0	EIDE
22	0x0	SDMMC4
21	0x0	USB2
20	0x0	USB
19	0x0	SDMMC3
18	0x0	AVP_UCQ
17	0x0	VDE
16	0x0	XIO
15	0x0	SDMMC2
14	0x0	SDMMC1
13	0x0	I2S1
12	0x0	VDE_SXE
11	0x0	VDE_BSEA
10	0x0	VDE_BSEV
9	0x0	VDE_SYNC_TOKEN
8	0x0	VDE_UCQ
7	0x0	SHR_SEM_OUTBOX_EMPTY
6	0x0	SHR_SEM_OUTBOX_FULL
5	0x0	SHR_SEM_INBOX_EMPTY
4	0x0	SHR_SEM_INBOX_FULL
3	0x0	I2S2

Bit	Reset	Description
2	0x0	RTC
1	0x0	TMR2
31:0	0x0	CPU_IER_SET: Writing a 1 in any bit position will enable the corresponding Interrupt Source for CPU
0	0x0	TMR1

3.2.1.11 PRI_ICTLR_CPU_IER_CLR_0

CPU Clear Interrupt Enable for CPU

Offset: 028h | Read/Write: WO | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	SDMMC4
30	0x0	OWR
29	0x0	ARB_SEM_GNT_CPU
28	0x0	ARB_SEM_GNT_COP
27	0x0	AHB_DMA_CPU
26	0x0	APB_DMA_CPU
25	0x0	VCP
24	0x0	NANDCTRL
23	0x0	EIDE
22	0x0	SDMMC4
21	0x0	USB2
20	0x0	USB
19	0x0	SDMMC3
18	0x0	AVP_UCQ
17	0x0	VDE
16	0x0	XIO
15	0x0	SDMMC2
14	0x0	SDMMC1
13	0x0	I2S1
12	0x0	VDE_SXE
11	0x0	VDE_BSEA
10	0x0	VDE_BSEV
9	0x0	VDE_SYNC_TOKEN

Bit	Reset	Description
8	0x0	VDE_UCQ
7	0x0	SHR_SEM_OUTBOX_EMPTY
6	0x0	SHR_SEM_OUTBOX_FULL
5	0x0	SHR_SEM_INBOX_EMPTY
4	0x0	SHR_SEM_INBOX_FULL
3	0x0	I2S2
2	0x0	RTC
1	0x0	TMR2
31:0	0x0	CPU_IER_CLR: Writing a 1 in any bit position will disable the corresponding Interrupt Source for CPU
0	0x0	TMR1

3.2.1.12 PRI_ICTLR_CPU_IEP_CLASS_0

CPU Interrupt Enable Priority Class (FIQ/IRQ)

Offset: 02ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	SDMMC4
30	0x0	OWR
29	0x0	ARB_SEM_GNT_CPU
28	0x0	ARB_SEM_GNT_COP
27	0x0	AHB_DMA_CPU
26	0x0	APB_DMA_CPU
25	0x0	VCP
24	0x0	NANDCTRL
23	0x0	EIDE
22	0x0	SDMMC4
21	0x0	USB2
20	0x0	USB
19	0x0	SDMMC3
18	0x0	AVP_UCQ
17	0x0	VDE
16	0x0	XIO
15	0x0	SDMMC2

Bit	Reset	Description
14	0x0	SDMMC1
13	0x0	I2S1
12	0x0	VDE_SXE
11	0x0	VDE_BSEA
10	0x0	VDE_BSEV
9	0x0	VDE_SYNC_TOKEN
8	0x0	VDE_UCQ
7	0x0	SHR_SEM_OUTBOX_EMPTY
6	0x0	SHR_SEM_OUTBOX_FULL
5	0x0	SHR_SEM_INBOX_EMPTY
4	0x0	SHR_SEM_INBOX_FULL
3	0x0	I2S2
2	0x0	RTC
1	0x0	TMR2
31:0	0x0	CPU_IEP_CLASS: Set Priority Interrupt Source For CPU. 1 = FIQ, 0 = IRQ.
0	0x0	TMR1

3.2.1.13 PRI_ICTLR_COP_IER_0

Enabled Interrupt Source for AVP (COP) Register

Offset: 030h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

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

Bit	Reset	Description
20	X	USB
19	X	SDMMC3
18	X	AVP_UCQ
17	X	VDE
16	X	XIO
15	X	SDMMC2
14	X	SDMMC1
13	X	I2S1
12	X	VDE_SXE
11	X	VDE_BSEA
10	X	VDE_BSEV
9	X	VDE_SYNC_TOKEN
8	X	VDE_UCQ
7	X	SHR_SEM_OUTBOX_EMPTY
6	X	SHR_SEM_OUTBOX_FULL
5	X	SHR_SEM_INBOX_EMPTY
4	X	SHR_SEM_INBOX_FULL
3	X	I2S2
2	X	RTC
1	X	TMR2
31:0	X	IER31_IER0: Interrupt Enable Status. 0 = Disabled
0	X	TMR1

3.2.1.14 PRI_ICTLR_COP_IER_SET_0

Set Interrupt Source for COP Register

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

Bit	Reset	Description
31	0x0	SDMMC4
30	0x0	OWR
29	0x0	ARB_SEM_GNT_CPU
28	0x0	ARB_SEM_GNT_COP
27	0x0	AHB_DMA_CPU

Bit	Reset	Description
26	0x0	APB_DMA_CPU
25	0x0	VCP
24	0x0	NANDCTRL
23	0x0	EIDE
22	0x0	SDMMC4
21	0x0	USB2
20	0x0	USB
19	0x0	SDMMC3
18	0x0	AVP_UCQ
17	0x0	VDE
16	0x0	XIO
15	0x0	SDMMC2
14	0x0	SDMMC1
13	0x0	I2S1
12	0x0	VDE_SXE
11	0x0	VDE_BSEA
10	0x0	VDE_BSEV
9	0x0	VDE_SYNC_TOKEN
8	0x0	VDE_UCQ
7	0x0	SHR_SEM_OUTBOX_EMPTY
6	0x0	SHR_SEM_OUTBOX_FULL
5	0x0	SHR_SEM_INBOX_EMPTY
4	0x0	SHR_SEM_INBOX_FULL
3	0x0	I2S2
2	0x0	RTC
1	0x0	TMR2
31:0	0x0	COP_IER_SET: Writing a 1 in any bit position will enable the corresponding Interrupt Source for COP.
0	0x0	TMR1

3.2.1.15 PRI_ICTLR_COP_IER_CLR_0

Clear Interrupt Source for COP Register

Offset: 038h | Read/Write: WO | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	SDMMC4
30	0x0	OWR
29	0x0	ARB_SEM_GNT_CPU
28	0x0	ARB_SEM_GNT_COP
27	0x0	AHB_DMA_CPU
26	0x0	APB_DMA_CPU
25	0x0	VCP
24	0x0	NANDCTRL
23	0x0	EIDE
22	0x0	SDMMC4
21	0x0	USB2
20	0x0	USB
19	0x0	SDMMC3
18	0x0	AVP_UCQ
17	0x0	VDE
16	0x0	XIO
15	0x0	SDMMC2
14	0x0	SDMMC1
13	0x0	I2S1
12	0x0	VDE_SXE
11	0x0	VDE_BSEA
10	0x0	VDE_BSEV
9	0x0	VDE_SYNC_TOKEN
8	0x0	VDE_UCQ
7	0x0	SHR_SEM_OUTBOX_EMPTY
6	0x0	SHR_SEM_OUTBOX_FULL
5	0x0	SHR_SEM_INBOX_EMPTY
4	0x0	SHR_SEM_INBOX_FULL
3	0x0	I2S2

Bit	Reset	Description
2	0x0	RTC
1	0x0	TMR2
31:0	0x0	COP_IER_CLR: Writing a 1 in any bit position will disable the corresponding Interrupt Source for COP
0	0x0	TMR1

3.2.1.16 PRI_ICTLR_COP_IEP_CLASS_0

COP Interrupt Enable Priority Class (FIQ/IRQ) Register

Offset: 03ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	SDMMC4
30	0x0	OWR
29	0x0	ARB_SEM_GNT_CPU
28	0x0	ARB_SEM_GNT_COP
27	0x0	AHB_DMA_CPU
26	0x0	APB_DMA_CPU
25	0x0	VCP
24	0x0	NANDCTRL
23	0x0	EIDE
22	0x0	SDMMC4
21	0x0	USB2
20	0x0	USB
19	0x0	SDMMC3
18	0x0	AVP_UCQ
17	0x0	VDE
16	0x0	XIO
15	0x0	SDMMC2
14	0x0	SDMMC1
13	0x0	I2S1
12	0x0	VDE_SXE
11	0x0	VDE_BSEA
10	0x0	VDE_BSEV
9	0x0	VDE_SYNC_TOKEN

Bit	Reset	Description
8	0x0	VDE_UCQ
7	0x0	SHR_SEM_OUTBOX_EMPTY
6	0x0	SHR_SEM_OUTBOX_FULL
5	0x0	SHR_SEM_INBOX_EMPTY
4	0x0	SHR_SEM_INBOX_FULL
3	0x0	I2S2
2	0x0	RTC
1	0x0	TMR2
31:0	0x0	COP_IEP_CLASS: Set Priority Interrupt Source For AVP (COP). 1 = FIQ 0 = IRQ
0	0x0	TMR1

3.2.2 Arbitration Semaphore Interrupt Registers

3.2.2.1 PRI_ICTLR_ARBGNT_CPU_STATUS_0

Arbitration semaphores provide a mechanism by which the two processors can arbitrate for the use of various resources. These semaphores provide a hardware locking mechanism, so that when a processor is already using a resource, the second processor is not granted that resource. There are 32 bits of Arbitration semaphores provided in the system. The hardware does not enforce any resource association to these bits. It is left to the firmware to assign and use these bits.

The Arbitration Semaphores can also generate an interrupt when a hardware resource becomes available. The registers in this module configure these interrupts.

When a 1 is set in the corresponding bit position of the Arbitration Semaphore Interrupt Source Register (CPU_enable or COP_enable), an interrupt will be generated when the processor achieves Grant Status for that resource.

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

CPU Arbitration Semaphore Interrupt Status Register

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

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

3.2.2.2 PRI_ICTLR_ARBGNT_CPU_ENABLE_0

CPU Arbitration Semaphore Interrupt Enable Register

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

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

3.2.2.3 PRI_ICTLR_ARBGNT_COP_STATUS_0

COP Arbitration Semaphore Interrupt Status Register

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

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

3.2.2.4 PRI_ICTLR_ARBGNT_COP_ENABLE_0

COP Arbitration Semaphore Interrupt Enable Register

Offset: 04ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

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

3.2.3 Second Interrupt Controller

3.2.3.1 SEC_ICTLR_VIRQ_CPU_0

Second interrupt controller base registers.

Valid Interrupt Request Status for CPU Register

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

Bit	Reset	Description
31	X	DMA_RX
30	X	DMA_TX
29	X	AHB_DMA_COP
28	X	APB_DMA_COP
27	X	SBC1
25	X	CPU1_PMU_INTR
24	X	CPU0_PMU_INTR
23	X	GPIO5

Bit	Reset	Description
22	X	STAT_MON
21	X	DVC
20	X	VFIR
19	X	EVENT_GPIO_D
18	X	EVENT_GPIO_C
17	X	EVENT_GPIO_B
16	X	EVENT_GPIO_A
15	X	MIPI_HS
14	X	UART3
13	X	SPDIF
12	X	FLOW_RSM_COP
11	X	FLOW_RSM_CPU
10	X	TMR4
9	X	TMR3
8	X	TWC
7	X	SPI
6	X	I2C
5	X	UART2
4	X	UART1
3	X	GPIO4
2	X	GPIO3
1	X	GPIO2
31:0	X	IRQ31_IRQ0: Flags set by Hardware, cleared by SW
0	X	GPIO1

3.2.3.2 SEC_ICTLR_VIRQ_COP_0

Valid Interrupt Status for COP Register

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

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

Bit	Reset	Description
25	X	CPU1_PMU_INTR
24	X	CPU0_PMU_INTR
23	X	GPIO5
22	X	STAT_MON
21	X	DVC
20	X	VFIR
19	X	EVENT_GPIO_D
18	X	EVENT_GPIO_C
17	X	EVENT_GPIO_B
16	X	EVENT_GPIO_A
15	X	MIPI_HS
14	X	UART3
13	X	SPDIF
12	X	FLOW_RSM_COP
11	X	FLOW_RSM_CPU
10	X	TMR4
9	X	TMR3
8	X	TWC
7	X	SPI
6	X	I2C
5	X	UART2
4	X	UART1
3	X	GPIO4
2	X	GPIO3
1	X	GPIO2
31:0	X	IRQ31_IRQ0: Flags set by Hardware, cleared by SW
0	X	GPIO1

3.2.3.3 SEC_ICTLR_VFIQ_CPU_0

FIQ Valid Interrupt Status for CPU Register

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

Bit	Reset	Description
31	X	DMA_RX
30	X	DMA_TX

Bit	Reset	Description
29	X	AHB_DMA_COP
28	X	APB_DMA_COP
27	X	SBC1
25	X	CPU1_PMU_INTR
24	X	CPU0_PMU_INTR
23	X	GPIO5
22	X	STAT_MON
21	X	DVC
20	X	VFIR
19	X	EVENT_GPIO_D
18	X	EVENT_GPIO_C
17	X	EVENT_GPIO_B
16	X	EVENT_GPIO_A
15	X	MIPI_HS
14	X	UART3
13	X	SPDIF
12	X	FLOW_RSM_COP
11	X	FLOW_RSM_CPU
10	X	TMR4
9	X	TMR3
8	X	TWC
7	X	SPI
6	X	I2C
5	X	UART2
4	X	UART1
3	X	GPIO4
2	X	GPIO3
1	X	GPIO2
31:0	X	FIQ31_FIQ0: Flags set by Hardware, cleared by SW
0	X	GPIO1

3.2.3.4 SEC_ICTLR_VFIQ_COP_0

FIQ Valid Interrupt Status for COP Register

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

Bit	Reset	Description
31	X	DMA_RX
30	X	DMA_TX
29	X	AHB_DMA_COP
28	X	APB_DMA_COP
27	X	SBC1
25	X	CPU1_PMU_INTR
24	X	CPU0_PMU_INTR
23	X	GPIO5
22	X	STAT_MON
21	X	DVC
20	X	VFIR
19	X	EVENT_GPIO_D
18	X	EVENT_GPIO_C
17	X	EVENT_GPIO_B
16	X	EVENT_GPIO_A
15	X	MIPI_HS
14	X	UART3
13	X	SPDIF
12	X	FLOW_RSM_COP
11	X	FLOW_RSM_CPU
10	X	TMR4
9	X	TMR3
8	X	TWC
7	X	SPI
6	X	I2C
5	X	UART2
4	X	UART1
3	X	GPIO4
2	X	GPIO3
1	X	GPIO2

Bit	Reset	Description
31:0	X	FIQ31_FIQ0: Flags set by Hardware, cleared by SW
0	X	GPIO1

3.2.3.5 SEC_ICTLR_ISR_0

Latched Interrupt Status Register (HW)

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

Bit	Reset	Description
31	X	DMA_RX
30	X	DMA_TX
29	X	AHB_DMA_COP
28	X	APB_DMA_COP
27	X	SBC1
25	X	CPU1_PMU_INTR
24	X	CPU0_PMU_INTR
23	X	GPIO5
22	X	STAT_MON
21	X	DVC
20	X	VFIR
19	X	EVENT_GPIO_D
18	X	EVENT_GPIO_C
17	X	EVENT_GPIO_B
16	X	EVENT_GPIO_A
15	X	MIPI_HS
14	X	UART3
13	X	SPDIF
12	X	FLOW_RSM_COP
11	X	FLOW_RSM_CPU
10	X	TMR4
9	X	TMR3
8	X	TWC
7	X	SPI
6	X	I2C
5	X	UART2
4	X	UART1

Bit	Reset	Description
3	X	GPIO4
2	X	GPIO3
1	X	GPIO2
31:0	X	ISR31_ISR0: Read-only. Set by hardware event, cleared at source by software.
0	X	GPIO1

3.2.3.6 SEC_ICTLR_FIR_0

Forced Interrupt Status Register (SW)

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

Bit	Reset	Description
31	X	DMA_RX
30	X	DMA_TX
29	X	AHB_DMA_COP
28	X	APB_DMA_COP
27	X	SBC1
25	X	CPU1_PMU_INTR
24	X	CPU0_PMU_INTR
23	X	GPIO5
22	X	STAT_MON
21	X	DVC
20	X	VFIR
19	X	EVENT_GPIO_D
18	X	EVENT_GPIO_C
17	X	EVENT_GPIO_B
16	X	EVENT_GPIO_A
15	X	MIPI_HS
14	X	UART3
13	X	SPDIF
12	X	FLOW_RSM_COP
11	X	FLOW_RSM_CPU
10	X	TMR4
9	X	TMR3
8	X	TWC
7	X	SPI

Bit	Reset	Description
6	X	I2C
5	X	UART2
4	X	UART1
3	X	GPIO4
2	X	GPIO3
1	X	GPIO2
31:0	X	FIR31_FIR0: Read only: Set during write to FIR_SET, cleared during write to FIR_CLR.
0	X	GPIO1

3.2.3.7 SEC_ICTLR_FIR_SET_0

Force Interrupt Register Set

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

Bit	Reset	Description
31	0x0	DMA_RX
30	0x0	DMA_TX
29	0x0	AHB_DMA_COP
28	0x0	APB_DMA_COP
27	0x0	SBC1
25	0x0	CPU1_PMU_INTR
24	0x0	CPU0_PMU_INTR
23	0x0	GPIO5
22	0x0	STAT_MON
21	0x0	DVC
20	0x0	VFIR
19	0x0	EVENT_GPIO_D
18	0x0	EVENT_GPIO_C
17	0x0	EVENT_GPIO_B
16	0x0	EVENT_GPIO_A
15	0x0	MIPI_HS
14	0x0	UART3
13	0x0	SPDIF
12	0x0	FLOW_RSM_COP
11	0x0	FLOW_RSM_CPU
10	0x0	TMR4

Bit	Reset	Description
9	0x0	TMR3
8	0x0	TWC
7	0x0	SPI
6	0x0	I2C
5	0x0	UART2
4	0x0	UART1
3	0x0	GPIO4
2	0x0	GPIO3
1	0x0	GPIO2
31:0	0x0	FIR_SET: Set Forced Interrupt Bit. Writing a 1 will set an interrupt
0	0x0	GPIO1

3.2.3.8 SEC_ICTLR_FIR_CLR_0

Force Interrupt Register Clear Register

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

Bit	Reset	Description
31	0x0	DMA_RX
30	0x0	DMA_TX
29	0x0	AHB_DMA_COP
28	0x0	APB_DMA_COP
27	0x0	SBC1
25	0x0	CPU1_PMU_INTR
24	0x0	CPU0_PMU_INTR
23	0x0	GPIO5
22	0x0	STAT_MON
21	0x0	DVC
20	0x0	VFIR
19	0x0	EVENT_GPIO_D
18	0x0	EVENT_GPIO_C
17	0x0	EVENT_GPIO_B
16	0x0	EVENT_GPIO_A
15	0x0	MIPI_HS
14	0x0	UART3
13	0x0	SPDIF

Bit	Reset	Description
12	0x0	FLOW_RSM_COP
11	0x0	FLOW_RSM_CPU
10	0x0	TMR4
9	0x0	TMR3
8	0x0	TWC
7	0x0	SPI
6	0x0	I2C
5	0x0	UART2
4	0x0	UART1
3	0x0	GPIO4
2	0x0	GPIO3
1	0x0	GPIO2
31:0	0x0	FIR_CLR: Clear Forced Interrupt Bit: Writing a 1 will clear the forced interrupt
0	0x0	GPIO1

3.2.3.9 SEC_ICTLR_CPU_IER_0

Enabled Interrupt Source for CPU Register

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

Bit	Reset	Description
31	X	DMA_RX
30	X	DMA_TX
29	X	AHB_DMA_COP
28	X	APB_DMA_COP
27	X	SBC1
25	X	CPU1_PMU_INTR
24	X	CPU0_PMU_INTR
23	X	GPIO5
22	X	STAT_MON
21	X	DVC
20	X	VFIR
19	X	EVENT_GPIO_D
18	X	EVENT_GPIO_C
17	X	EVENT_GPIO_B
16	X	EVENT_GPIO_A

Bit	Reset	Description
15	X	MIPI_HS
14	X	UART3
13	X	SPDIF
12	X	FLOW_RSM_COP
11	X	FLOW_RSM_CPU
10	X	TMR4
9	X	TMR3
8	X	TWC
7	X	SPI
6	X	I2C
5	X	UART2
4	X	UART1
3	X	GPIO4
2	X	GPIO3
1	X	GPIO2
31:0	X	IER31_IER0: Interrupt Enable Status. 0 = Disabled
0	X	GPIO1

3.2.3.10 SEC_ICTLR_CPU_IER_SET_0

Set Interrupt Enable for CPU Register

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

Bit	Reset	Description
31	0x0	DMA_RX
30	0x0	DMA_TX
29	0x0	AHB_DMA_COP
28	0x0	APB_DMA_COP
27	0x0	SBC1
25	0x0	CPU1_PMU_INTR
24	0x0	CPU0_PMU_INTR
23	0x0	GPIO5
22	0x0	STAT_MON
21	0x0	DVC
20	0x0	VFIR
19	0x0	EVENT_GPIO_D

Bit	Reset	Description
18	0x0	EVENT_GPIO_C
17	0x0	EVENT_GPIO_B
16	0x0	EVENT_GPIO_A
15	0x0	MIPI_HS
14	0x0	UART3
13	0x0	SPDIF
12	0x0	FLOW_RSM_COP
11	0x0	FLOW_RSM_CPU
10	0x0	TMR4
9	0x0	TMR3
8	0x0	TWC
7	0x0	SPI
6	0x0	I2C
5	0x0	UART2
4	0x0	UART1
3	0x0	GPIO4
2	0x0	GPIO3
1	0x0	GPIO2
31:0	0x0	CPU_IER_SET: Writing a 1 in any bit position will enable the corresponding Interrupt Source for CPU
0	0x0	GPIO1

3.2.3.11 SEC_ICTLR_CPU_IER_CLR_0

CPUs Clear Interrupt Enable for CPU

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

Bit	Reset	Description
31	0x0	DMA_RX
30	0x0	DMA_TX
29	0x0	AHB_DMA_COP
28	0x0	APB_DMA_COP
27	0x0	SBC1
25	0x0	CPU1_PMU_INTR
24	0x0	CPU0_PMU_INTR
23	0x0	GPIO5

Bit	Reset	Description
22	0x0	STAT_MON
21	0x0	DVC
20	0x0	VFIR
19	0x0	EVENT_GPIO_D
18	0x0	EVENT_GPIO_C
17	0x0	EVENT_GPIO_B
16	0x0	EVENT_GPIO_A
15	0x0	MIPI_HS
14	0x0	UART3
13	0x0	SPDIF
12	0x0	FLOW_RSM_COP
11	0x0	FLOW_RSM_CPU
10	0x0	TMR4
9	0x0	TMR3
8	0x0	TWC
7	0x0	SPI
6	0x0	I2C
5	0x0	UART2
4	0x0	UART1
3	0x0	GPIO4
2	0x0	GPIO3
1	0x0	GPIO2
31:0	0x0	CPU_IER_CLR: Writing a 1 in any bit position will disable the corresponding Interrupt Source for CPU
0	0x0	GPIO1

3.2.3.12 SEC_ICTLR_CPU_IEP_CLASS_0

CPUs Interrupt Enable Priority Class (FIQ/IRQ)

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

Bit	Reset	Description
31	0x0	DMA_RX
30	0x0	DMA_TX
29	0x0	AHB_DMA_COP
28	0x0	APB_DMA_COP

Bit	Reset	Description
27	0x0	SBC1
25	0x0	CPU1_PMU_INTR
24	0x0	CPU0_PMU_INTR
23	0x0	GPIO5
22	0x0	STAT_MON
21	0x0	DVC
20	0x0	VFIR
19	0x0	EVENT_GPIO_D
18	0x0	EVENT_GPIO_C
17	0x0	EVENT_GPIO_B
16	0x0	EVENT_GPIO_A
15	0x0	MIPI_HS
14	0x0	UART3
13	0x0	SPDIF
12	0x0	FLOW_RSM_COP
11	0x0	FLOW_RSM_CPU
10	0x0	TMR4
9	0x0	TMR3
8	0x0	TWC
7	0x0	SPI
6	0x0	I2C
5	0x0	UART2
4	0x0	UART1
3	0x0	GPIO4
2	0x0	GPIO3
1	0x0	GPIO2
31:0	0x0	CPU_IEP_CLASS: Set Priority Interrupt Source For CPU. 1 = FIQ, 0 = IRQ.
0	0x0	GPIO1

3.2.3.13 SEC_ICTLR_COP_IER_0

Enabled Interrupt Source for COP Register

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

Bit	Reset	Description
31	X	DMA_RX

Bit	Reset	Description
30	X	DMA_TX
29	X	AHB_DMA_COP
28	X	APB_DMA_COP
27	X	SBC1
25	X	CPU1_PMU_INTR
24	X	CPU0_PMU_INTR
23	X	GPIO5
22	X	STAT_MON
21	X	DVC
20	X	VFIR
19	X	EVENT_GPIO_D
18	X	EVENT_GPIO_C
17	X	EVENT_GPIO_B
16	X	EVENT_GPIO_A
15	X	MIPI_HS
14	X	UART3
13	X	SPDIF
12	X	FLOW_RSM_COP
11	X	FLOW_RSM_CPU
10	X	TMR4
9	X	TMR3
8	X	TWC
7	X	SPI
6	X	I2C
5	X	UART2
4	X	UART1
3	X	GPIO4
2	X	GPIO3
1	X	GPIO2
31:0	X	IER31_IER0: Interrupt Enable Status. 0 = Disabled.
0	X	GPIO1

3.2.3.14 SEC_ICTLR_COP_IER_SET_0

Set Interrupt Source for COP Register

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

Bit	Reset	Description
31	0x0	DMA_RX
30	0x0	DMA_TX
29	0x0	AHB_DMA_COP
28	0x0	APB_DMA_COP
27	0x0	SBC1
25	0x0	CPU1_PMU_INTR
24	0x0	CPU0_PMU_INTR
23	0x0	GPIO5
22	0x0	STAT_MON
21	0x0	DVC
20	0x0	VFIR
19	0x0	EVENT_GPIO_D
18	0x0	EVENT_GPIO_C
17	0x0	EVENT_GPIO_B
16	0x0	EVENT_GPIO_A
15	0x0	MIPI_HS
14	0x0	UART3
13	0x0	SPDIF
12	0x0	FLOW_RSM_COP
11	0x0	FLOW_RSM_CPU
10	0x0	TMR4
9	0x0	TMR3
8	0x0	TWC
7	0x0	SPI
6	0x0	I2C
5	0x0	UART2
4	0x0	UART1
3	0x0	GPIO4
2	0x0	GPIO3
1	0x0	GPIO2

Bit	Reset	Description
31:0	0x0	COP_IER_SET: Writing a 1 in any bit position will enable the corresponding Interrupt Source for COP
0	0x0	GPIO1

3.2.3.15 SEC_ICTLR_COP_IER_CLR_0

Clear Interrupt Source for COP Register

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

Bit	Reset	Description
31	0x0	DMA_RX
30	0x0	DMA_TX
29	0x0	AHB_DMA_COP
28	0x0	APB_DMA_COP
27	0x0	SBC1
25	0x0	CPU1_PMU_INTR
24	0x0	CPU0_PMU_INTR
23	0x0	GPIO5
22	0x0	STAT_MON
21	0x0	DVC
20	0x0	VFIR
19	0x0	EVENT_GPIO_D
18	0x0	EVENT_GPIO_C
17	0x0	EVENT_GPIO_B
16	0x0	EVENT_GPIO_A
15	0x0	MIPI_HS
14	0x0	UART3
13	0x0	SPDIF
12	0x0	FLOW_RSM_COP
11	0x0	FLOW_RSM_CPU
10	0x0	TMR4
9	0x0	TMR3
8	0x0	TWC
7	0x0	SPI
6	0x0	I2C
5	0x0	UART2

Bit	Reset	Description
4	0x0	UART1
3	0x0	GPIO4
2	0x0	GPIO3
1	0x0	GPIO2
31:0	0x0	COP_IER_CLR: Writing a 1 in any bit position will disable the corresponding Interrupt Source for COP
0	0x0	GPIO1

3.2.3.16 SEC_ICTLR_COP_IEP_CLASS_0

COPs Interrupt Enable Priority Class (FIQ/IRQ) Register

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

Bit	Reset	Description
31	0x0	DMA_RX
30	0x0	DMA_TX
29	0x0	AHB_DMA_COP
28	0x0	APB_DMA_COP
27	0x0	SBC1
25	0x0	CPU1_PMU_INTR
24	0x0	CPU0_PMU_INTR
23	0x0	GPIO5
22	0x0	STAT_MON
21	0x0	DVC
20	0x0	VFIR
19	0x0	EVENT_GPIO_D
18	0x0	EVENT_GPIO_C
17	0x0	EVENT_GPIO_B
16	0x0	EVENT_GPIO_A
15	0x0	MIPI_HS
14	0x0	UART3
13	0x0	SPDIF
12	0x0	FLOW_RSM_COP
11	0x0	FLOW_RSM_CPU
10	0x0	TMR4
9	0x0	TMR3

Bit	Reset	Description
8	0x0	TWC
7	0x0	SPI
6	0x0	I2C
5	0x0	UART2
4	0x0	UART1
3	0x0	GPIO4
2	0x0	GPIO3
1	0x0	GPIO2
31:0	0x0	COP_IEP_CLASS: Set Priority Interrupt Source For COP. 1 = FIQ, 0 = IRQ.
0	0x0	GPIO1

3.2.3.17 SEC_ICTLR_DRQ_TX_STATUS_0

Second Interrupt Controller DRQ TX Registers.

DRQ Interrupt Source Status

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

Bit	Reset	Description
18	X	AC97_MPB
17	X	VFIR_TX_FIFO
16	X	I2C3_TX
15	X	UART5_OUTPUT_FIFO
14	X	UART4_OUTPUT_FIFO
13	X	I2C2_TX
12	X	I2C_TX
11	X	TWC_TX
10	X	UART3_OUTPUT_FIFO
9	X	UART2_OUTPUT_FIFO
8	X	UART1_OUTPUT_FIFO
7	X	SPI_TX
6	X	I2S2_TX_FIFO1
5	X	I2S2_TX_FIFO2
4	X	SPDIF_USER_TX
3	X	SPDIF_DATA_TX
2	X	I2S1_TX_FIFO1
1	X	I2S1_TX_FIFO2

Bit	Reset	Description
31:0	X	DRQ31_DRQ0: DRQ_INTERRUPT_SOURCE_STATUS
0	X	AC97_PB

3.2.3.18 SEC_ICTLR_DRQ_TX_ENABLE_0

DRQ Interrupt Source Enable Gates

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

Bit	R/W	Reset	Description
18	RO	X	AC97_MPB
17	RO	X	VFIR_TX_FIFO
16	RO	X	I2C3_TX
15	RO	X	UART5_OUTPUT_FIFO
14	RO	X	UART4_OUTPUT_FIFO
13	RO	X	I2C2_TX
12	RO	X	I2C_TX
11	RO	X	TWC_TX
10	RO	X	UART3_OUTPUT_FIFO
9	RO	X	UART2_OUTPUT_FIFO
8	RO	X	UART1_OUTPUT_FIFO
7	RO	X	SPI_TX
6	RO	X	I2S2_TX_FIFO1
5	RO	X	I2S2_TX_FIFO2
4	RO	X	SPDIF_USER_TX
3	RO	X	SPDIF_DATA_TX
2	RO	X	I2S1_TX_FIFO1
1	RO	X	I2S1_TX_FIFO2
31:0	RO	0x0	DER31_DER0: DRQ_INTERRUPT_SOURCE_ENABLE_GATES
0	RO	X	AC97_PB

3.2.3.19 SEC_ICTLR_DRQ_RX_STATUS_0

Second interrupt controller DRQ RX registers.

DRQ Interrupt Source Status por=0x00000000

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

Bit	Reset	Description
18	X	AC97_MREC

Bit	Reset	Description
17	X	VFIR_RX_FIFO
16	X	I2C3_RX
15	X	UART5_INPUT_FIFO
14	X	UART4_INPUT_FIFO
13	X	I2C2_RX
12	X	I2C_RX
11	X	TWC_RX
10	X	UART3_INPUT_FIFO
9	X	UART2_INPUT_FIFO
8	X	UART1_INPUT_FIFO
7	X	SPI_RX
6	X	I2S2_RX_FIFO2
5	X	I2S2_RX_FIFO1
4	X	SPDIF_USER_RX
3	X	SPDIF_DATA_RX
2	X	I2S1_RX_FIFO2
1	X	I2S1_RX_FIFO1
31:0	X	DRQ31_DRQ0: DRQ_INTERRUPT_SOURCE_STATUS
0	X	AC97_REC

3.2.3.20 SEC_ICTLR_DRQ_RX_ENABLE_0

DRQ Interrupt Source Enable Gates

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

Bit	R/W	Reset	Description
18	RO	X	AC97_MREC
17	RO	X	VFIR_RX_FIFO
16	RO	X	I2C3_RX
15	RO	X	UART5_INPUT_FIFO
14	RO	X	UART4_INPUT_FIFO
13	RO	X	I2C2_RX
12	RO	X	I2C_RX
11	RO	X	TWC_RX
10	RO	X	UART3_INPUT_FIFO
9	RO	X	UART2_INPUT_FIFO

Bit	R/W	Reset	Description
8	RO	X	UART1_INPUT_FIFO
7	RO	X	SPI_RX
6	RO	X	I2S2_RX_FIFO2
5	RO	X	I2S2_RX_FIFO1
4	RO	X	SPDIF_USER_RX
3	RO	X	SPDIF_DATA_RX
2	RO	X	I2S1_RX_FIFO2
1	RO	X	I2S1_RX_FIFO1
31:0	RO	0x0	DER31_DER0: DRQ_INTERRUPT_SOURCE_ENABLE_GATES
0	RO	X	AC97_REC

3.2.4 Third Interrupt Controller Registers

3.2.4.1 TRI_ICTLR_VIRQ_CPU_0

Third interrupt controller base registers.

Valid Interrupt Request Status for CPU Register

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

Bit	Reset	Description
31	X	SW_INTR
29	X	SBC4
28	X	I2C3
27	X	UART5
26	X	UART4
25	X	GPIO7
24	X	TVDAC
23	X	GPIO6
22	X	PMU_EXT
21	X	KBC
20	X	I2C2
19	X	SBC3
18	X	SBC2
17	X	AC97
16	X	NOR_FLASH
14	X	EMC

Bit	Reset	Description
13	X	MC
12	X	TVO
11	X	HDMI
10	X	DISPLAYB
9	X	DISPLAY
8	X	GR2D
7	X	ISP
6	X	EPP
5	X	VI
4	X	MPE
3	X	HOST1X_GEN_CPU
2	X	HOST1X_GEN_COP
1	X	HOST1X_SYNCPT_CPU
31:0	X	IRQ31_IRQ0: Flags set by Hardware, cleared by SW
0	X	HOST1X_SYNCPT_COP

3.2.4.2 TRI_ICTLR_VIRQ_COP_0

Valid Interrupt Status for COP Register

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

Bit	Reset	Description
31	X	SW_INTR
29	X	SBC4
28	X	I2C3
27	X	UART5
26	X	UART4
25	X	GPIO7
24	X	TVDAC
23	X	GPIO6
22	X	PMU_EXT
21	X	KBC
20	X	I2C2
19	X	SBC3
18	X	SBC2
17	X	AC97

Bit	Reset	Description
16	X	NOR_FLASH
14	X	EMC
13	X	MC
12	X	TVO
11	X	HDMI
10	X	DISPLAYB
9	X	DISPLAY
8	X	GR2D
7	X	ISP
6	X	EPP
5	X	VI
4	X	MPE
3	X	HOST1X_GEN_CPU
2	X	HOST1X_GEN_COP
1	X	HOST1X_SYNCPT_CPU
31:0	X	IRQ31_IRQ0: Flags set by Hardware, cleared by SW
0	X	HOST1X_SYNCPT_COP

3.2.4.3 TRI_ICTLR_VFIQ_CPU_0

FIQ Valid Interrupt Status for CPU Register

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

Bit	Reset	Description
31	X	SW_INTR
29	X	SBC4
28	X	I2C3
27	X	UART5
26	X	UART4
25	X	GPIO7
24	X	TVDAC
23	X	GPIO6
22	X	PMU_EXT
21	X	KBC
20	X	I2C2
19	X	SBC3

Bit	Reset	Description
18	X	SBC2
17	X	AC97
16	X	NOR_FLASH
14	X	EMC
13	X	MC
12	X	TVO
11	X	HDMI
10	X	DISPLAYB
9	X	DISPLAY
8	X	GR2D
7	X	ISP
6	X	EPP
5	X	VI
4	X	MPE
3	X	HOST1X_GEN_CPU
2	X	HOST1X_GEN_COP
1	X	HOST1X_SYNCPT_CPU
31:0	X	FIQ31_FIQ0: Flags set by Hardware, cleared by SW
0	X	HOST1X_SYNCPT_COP

3.2.4.4 TRI_ICTLR_VFIQ_COP_0

FIQ Valid Interrupt Status for COP Register

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

Bit	Reset	Description
31	X	SW_INTR
29	X	SBC4
28	X	I2C3
27	X	UART5
26	X	UART4
25	X	GPIO7
24	X	TVDAC
23	X	GPIO6
22	X	PMU_EXT
21	X	KBC

Bit	Reset	Description
20	X	I2C2
19	X	SBC3
18	X	SBC2
17	X	AC97
16	X	NOR_FLASH
14	X	EMC
13	X	MC
12	X	TVO
11	X	HDMI
10	X	DISPLAYB
9	X	DISPLAY
8	X	GR2D
7	X	ISP
6	X	EPP
5	X	VI
4	X	MPE
3	X	HOST1X_GEN_CPU
2	X	HOST1X_GEN_COP
1	X	HOST1X_SYNCPT_CPU
31:0	X	FIQ31_FIQ0: Flags set by Hardware, cleared by SW
0	X	HOST1X_SYNCPT_COP

3.2.4.5 TRI_ICTLR_ISR_0

Latched Interrupt Status Register (HW)

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

Bit	Reset	Description
31	X	SW_INTR
29	X	SBC4
28	X	I2C3
27	X	UART5
26	X	UART4
25	X	GPIO7
24	X	TVDAC
23	X	GPIO6

Bit	Reset	Description
22	X	PMU_EXT
21	X	KBC
20	X	I2C2
19	X	SBC3
18	X	SBC2
17	X	AC97
16	X	NOR_FLASH
14	X	EMC
13	X	MC
12	X	TVO
11	X	HDMI
10	X	DISPLAYB
9	X	DISPLAY
8	X	GR2D
7	X	ISP
6	X	EPP
5	X	VI
4	X	MPE
3	X	HOST1X_GEN_CPU
2	X	HOST1X_GEN_COP
1	X	HOST1X_SYNCPT_CPU
31:0	X	ISR31_ISR0: Read-only. Set by hardware event, cleared at source by software.
0	X	HOST1X_SYNCPT_COP

3.2.4.6 TRI_ICTLR_FIR_0

Forced Interrupt Status Register (SW)

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

Bit	Reset	Description
31	X	SW_INTR
29	X	SBC4
28	X	I2C3
27	X	UART5
26	X	UART4
25	X	GPIO7

Bit	Reset	Description
24	X	TVDAC
23	X	GPIO6
22	X	PMU_EXT
21	X	KBC
20	X	I2C2
19	X	SBC3
18	X	SBC2
17	X	AC97
16	X	NOR_FLASH
14	X	EMC
13	X	MC
12	X	TVO
11	X	HDMI
10	X	DISPLAYB
9	X	DISPLAY
8	X	GR2D
7	X	ISP
6	X	EPP
5	X	VI
4	X	MPE
3	X	HOST1X_GEN_CPU
2	X	HOST1X_GEN_COP
1	X	HOST1X_SYNCPT_CPU
31:0	X	FIR31_FIR0: Read only: Set during write to FIR_SET, cleared during write to FIR_CLR.
0	X	HOST1X_SYNCPT_COP

3.2.4.7 TRI_ICTLR_FIR_SET_0

Force Interrupt Register Set

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

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

Bit	Reset	Description
27	0x0	UART5
26	0x0	UART4
25	0x0	GPIO7
24	0x0	TVDAC
23	0x0	GPIO6
22	0x0	PMU_EXT
21	0x0	KBC
20	0x0	I2C2
19	0x0	SBC3
18	0x0	SBC2
17	0x0	AC97
16	0x0	NOR_FLASH
14	0x0	EMC
13	0x0	MC
12	0x0	TVO
11	0x0	HDMI
10	0x0	DISPLAYB
9	0x0	DISPLAY
8	0x0	GR2D
7	0x0	ISP
6	0x0	EPP
5	0x0	VI
4	0x0	MPE
3	0x0	HOST1X_GEN_CPU
2	0x0	HOST1X_GEN_COP
1	0x0	HOST1X_SYNCPT_CPU
31:0	0x0	FIR_SET: Set Forced Interrupt Bit. Writing a 1 will set an interrupt
0	0x0	HOST1X_SYNCPT_COP

3.2.4.8 TRI_ICTLR_FIR_CLR_0

Force Interrupt Register Clear Register

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

Bit	Reset	Description
31	0x0	SW_INTR

Bit	Reset	Description
29	0x0	SBC4
28	0x0	I2C3
27	0x0	UART5
26	0x0	UART4
25	0x0	GPIO7
24	0x0	TVDAC
23	0x0	GPIO6
22	0x0	PMU_EXT
21	0x0	KBC
20	0x0	I2C2
19	0x0	SBC3
18	0x0	SBC2
17	0x0	AC97
16	0x0	NOR_FLASH
14	0x0	EMC
13	0x0	MC
12	0x0	TVO
11	0x0	HDMI
10	0x0	DISPLAYB
9	0x0	DISPLAY
8	0x0	GR2D
7	0x0	ISP
6	0x0	EPP
5	0x0	VI
4	0x0	MPE
3	0x0	HOST1X_GEN_CPU
2	0x0	HOST1X_GEN_COP
1	0x0	HOST1X_SYNCPT_CPU
31:0	0x0	FIR_CLR: Clear Forced Interrupt Bit: Writing a 1 will clear the forced interrupt
0	0x0	HOST1X_SYNCPT_COP

3.2.4.9 TRI_ICTLR_CPU_IER_0

Enabled Interrupt Source for CPU Register

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

Bit	Reset	Description
31	X	SW_INTR
29	X	SBC4
28	X	I2C3
27	X	UART5
26	X	UART4
25	X	GPIO7
24	X	TVDAC
23	X	GPIO6
22	X	PMU_EXT
21	X	KBC
20	X	I2C2
19	X	SBC3
18	X	SBC2
17	X	AC97
16	X	NOR_FLASH
14	X	EMC
13	X	MC
12	X	TVO
11	X	HDMI
10	X	DISPLAYB
9	X	DISPLAY
8	X	GR2D
7	X	ISP
6	X	EPP
5	X	VI
4	X	MPE
3	X	HOST1X_GEN_CPU
2	X	HOST1X_GEN_COP
1	X	HOST1X_SYNCPT_CPU
31:0	X	IER31_IER0: Interrupt Enable Status. 0 = Disabled

Bit	Reset	Description
0	X	HOST1X_SYNCPT_COP

3.2.4.10 TRI_ICTLR_CPU_IER_SET_0

Set Interrupt Enable for CPU Register

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

Bit	Reset	Description
31	0x0	SW_INTR
29	0x0	SBC4
28	0x0	I2C3
27	0x0	UART5
26	0x0	UART4
25	0x0	GPIO7
24	0x0	TVDAC
23	0x0	GPIO6
22	0x0	PMU_EXT
21	0x0	KBC
20	0x0	I2C2
19	0x0	SBC3
18	0x0	SBC2
17	0x0	AC97
16	0x0	NOR_FLASH
14	0x0	EMC
13	0x0	MC
12	0x0	TVO
11	0x0	HDMI
10	0x0	DISPLAYB
9	0x0	DISPLAY
8	0x0	GR2D
7	0x0	ISP
6	0x0	EPP
5	0x0	VI
4	0x0	MPE
3	0x0	HOST1X_GEN_CPU
2	0x0	HOST1X_GEN_COP

Bit	Reset	Description
1	0x0	HOST1X_SYNCPT_CPU
31:0	0x0	CPU_IER_SET: Writing a 1 in any bit position will enable the corresponding Interrupt Source for CPU
0	0x0	HOST1X_SYNCPT_COP

3.2.4.11 TRI_ICTLR_CPU_IER_CLR_0

CPUs Clear Interrupt Enable for CPU

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

Bit	Reset	Description
31	0x0	SW_INTR
29	0x0	SBC4
28	0x0	I2C3
27	0x0	UART5
26	0x0	UART4
25	0x0	GPIO7
24	0x0	TVDAC
23	0x0	GPIO6
22	0x0	PMU_EXT
21	0x0	KBC
20	0x0	I2C2
19	0x0	SBC3
18	0x0	SBC2
17	0x0	AC97
16	0x0	NOR_FLASH
14	0x0	EMC
13	0x0	MC
12	0x0	TVO
11	0x0	HDMI
10	0x0	DISPLAYB
9	0x0	DISPLAY
8	0x0	GR2D
7	0x0	ISP
6	0x0	EPP
5	0x0	VI

Bit	Reset	Description
4	0x0	MPE
3	0x0	HOST1X_GEN_CPU
2	0x0	HOST1X_GEN_COP
1	0x0	HOST1X_SYNCPT_CPU
31:0	0x0	CPU_IER_CLR: Writing a 1 in any bit position will disable the corresponding Interrupt Source for CPU
0	0x0	HOST1X_SYNCPT_COP

3.2.4.12 TRI_ICTLR_CPU_IEP_CLASS_0

CPUs Interrupt Enable Priority Class (FIQ/IRQ)

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

Bit	Reset	Description
31	0x0	SW_INTR
29	0x0	SBC4
28	0x0	I2C3
27	0x0	UART5
26	0x0	UART4
25	0x0	GPIO7
24	0x0	TVDAC
23	0x0	GPIO6
22	0x0	PMU_EXT
21	0x0	KBC
20	0x0	I2C2
19	0x0	SBC3
18	0x0	SBC2
17	0x0	AC97
16	0x0	NOR_FLASH
14	0x0	EMC
13	0x0	MC
12	0x0	TVO
11	0x0	HDMI
10	0x0	DISPLAYB
9	0x0	DISPLAY
8	0x0	GR2D

Bit	Reset	Description
7	0x0	ISP
6	0x0	EPP
5	0x0	VI
4	0x0	MPE
3	0x0	HOST1X_GEN_CPU
2	0x0	HOST1X_GEN_COP
1	0x0	HOST1X_SYNCPT_CPU
31:0	0x0	CPU_IEP_CLASS: Set Priority Interrupt Source For CPU. 1 = FIQ, 0 = IRQ.
0	0x0	HOST1X_SYNCPT_COP

3.2.4.13 TRI_ICTLR_COP_IER_0

Enabled Interrupt Source for COP Register

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

Bit	Reset	Description
31	X	SW_INTR
29	X	SBC4
28	X	I2C3
27	X	UART5
26	X	UART4
25	X	GPIO7
24	X	TVDAC
23	X	GPIO6
22	X	PMU_EXT
21	X	KBC
20	X	I2C2
19	X	SBC3
18	X	SBC2
17	X	AC97
16	X	NOR_FLASH
14	X	EMC
13	X	MC
12	X	TVO
11	X	HDMI
10	X	DISPLAYB

Bit	Reset	Description
9	X	DISPLAY
8	X	GR2D
7	X	ISP
6	X	EPP
5	X	VI
4	X	MPE
3	X	HOST1X_GEN_CPU
2	X	HOST1X_GEN_COP
1	X	HOST1X_SYNCPT_CPU
31:0	X	IER31_IER0: Interrupt Enable Status. 0 = Disabled.
0	X	HOST1X_SYNCPT_COP

3.2.4.14 TRI_ICTLR_COP_IER_SET_0

Set Interrupt Source for COP Register

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

Bit	Reset	Description
31	0x0	SW_INTR
29	0x0	SBC4
28	0x0	I2C3
27	0x0	UART5
26	0x0	UART4
25	0x0	GPIO7
24	0x0	TVDAC
23	0x0	GPIO6
22	0x0	PMU_EXT
21	0x0	KBC
20	0x0	I2C2
19	0x0	SBC3
18	0x0	SBC2
17	0x0	AC97
16	0x0	NOR_FLASH
14	0x0	EMC
13	0x0	MC
12	0x0	TVO

Bit	Reset	Description
11	0x0	HDMI
10	0x0	DISPLAYB
9	0x0	DISPLAY
8	0x0	GR2D
7	0x0	ISP
6	0x0	EPP
5	0x0	VI
4	0x0	MPE
3	0x0	HOST1X_GEN_CPU
2	0x0	HOST1X_GEN_COP
1	0x0	HOST1X_SYNCPT_CPU
31:0	0x0	COP_IER_SET: Writing a 1 in any bit position will enable the corresponding Interrupt Source for COP
0	0x0	HOST1X_SYNCPT_COP

3.2.4.15 TRI_ICTLR_COP_IER_CLR_0

Clear Interrupt Source for COP Register

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

Bit	Reset	Description
31	0x0	SW_INTR
29	0x0	SBC4
28	0x0	I2C3
27	0x0	UART5
26	0x0	UART4
25	0x0	GPIO7
24	0x0	TVDAC
23	0x0	GPIO6
22	0x0	PMU_EXT
21	0x0	KBC
20	0x0	I2C2
19	0x0	SBC3
18	0x0	SBC2
17	0x0	AC97
16	0x0	NOR_FLASH

Bit	Reset	Description
14	0x0	EMC
13	0x0	MC
12	0x0	TVO
11	0x0	HDMI
10	0x0	DISPLAYB
9	0x0	DISPLAY
8	0x0	GR2D
7	0x0	ISP
6	0x0	EPP
5	0x0	VI
4	0x0	MPE
3	0x0	HOST1X_GEN_CPU
2	0x0	HOST1X_GEN_COP
1	0x0	HOST1X_SYNCPT_CPU
31:0	0x0	COP_IER_CLR: Writing a 1 in any bit position will disable the corresponding Interrupt Source for COP
0	0x0	HOST1X_SYNCPT_COP

3.2.4.16 TRI_ICTLR_COP_IEP_CLASS_0

COPs Interrupt Enable Priority Class (FIQ/IRQ) Register

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

Bit	Reset	Description
31	0x0	SW_INTR
29	0x0	SBC4
28	0x0	I2C3
27	0x0	UART5
26	0x0	UART4
25	0x0	GPIO7
24	0x0	TVDAC
23	0x0	GPIO6
22	0x0	PMU_EXT
21	0x0	KBC
20	0x0	I2C2
19	0x0	SBC3

Bit	Reset	Description
18	0x0	SBC2
17	0x0	AC97
16	0x0	NOR_FLASH
14	0x0	EMC
13	0x0	MC
12	0x0	TVO
11	0x0	HDMI
10	0x0	DISPLAYB
9	0x0	DISPLAY
8	0x0	GR2D
7	0x0	ISP
6	0x0	EPP
5	0x0	VI
4	0x0	MPE
3	0x0	HOST1X_GEN_CPU
2	0x0	HOST1X_GEN_COP
1	0x0	HOST1X_SYNCPT_CPU
31:0	0x0	COP_IEP_CLASS: Set Priority Interrupt Source For COP. 1 = FIQ, 0 = IRQ.
0	0x0	HOST1X_SYNCPT_COP

3.2.5 Fourth Interrupt Controller Registers

3.2.5.1 QUAD_ICTLR_VIRQ_CPU_0

Fourth interrupt controller base registers

Valid Interrupt Request Status for CPU Register

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

Bit	Reset	Description
23	X	APB_DMA_CH15
22	X	APB_DMA_CH14
21	X	APB_DMA_CH13
20	X	APB_DMA_CH12
19	X	APB_DMA_CH11
18	X	APB_DMA_CH10
17	X	APB_DMA_CH9

Bit	Reset	Description
16	X	APB_DMA_CH8
15	X	APB_DMA_CH7
14	X	APB_DMA_CH6
13	X	APB_DMA_CH5
12	X	APB_DMA_CH4
11	X	APB_DMA_CH3
10	X	APB_DMA_CH2
9	X	APB_DMA_CH1
8	X	APB_DMA_CH0
4	X	PCIE_WAKE
3	X	PCIE_MSI
2	X	PCIE_INT
1	X	USB3
31:0	X	IRQ31_IRQ0: Flags set by Hardware, cleared by SW
0	X	SNOR

3.2.5.2 QUAD_ICTLR_VIRQ_COP_0

Valid Interrupt Status for COP Register

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

Bit	Reset	Description
23	X	APB_DMA_CH15
22	X	APB_DMA_CH14
21	X	APB_DMA_CH13
20	X	APB_DMA_CH12
19	X	APB_DMA_CH11
18	X	APB_DMA_CH10
17	X	APB_DMA_CH9
16	X	APB_DMA_CH8
15	X	APB_DMA_CH7
14	X	APB_DMA_CH6
13	X	APB_DMA_CH5
12	X	APB_DMA_CH4
11	X	APB_DMA_CH3
10	X	APB_DMA_CH2

Bit	Reset	Description
9	X	APB_DMA_CH1
8	X	APB_DMA_CH0
4	X	PCIE_WAKE
3	X	PCIE_MSI
2	X	PCIE_INT
1	X	USB3
31:0	X	IRQ31_IRQ0: Flags set by Hardware, cleared by SW
0	X	SNOR

3.2.5.3 QUAD_ICTLR_VFIQ_CPU_0

FIQ Valid Interrupt Status for CPU Register

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

Bit	Reset	Description
23	X	APB_DMA_CH15
22	X	APB_DMA_CH14
21	X	APB_DMA_CH13
20	X	APB_DMA_CH12
19	X	APB_DMA_CH11
18	X	APB_DMA_CH10
17	X	APB_DMA_CH9
16	X	APB_DMA_CH8
15	X	APB_DMA_CH7
14	X	APB_DMA_CH6
13	X	APB_DMA_CH5
12	X	APB_DMA_CH4
11	X	APB_DMA_CH3
10	X	APB_DMA_CH2
9	X	APB_DMA_CH1
8	X	APB_DMA_CH0
4	X	PCIE_WAKE
3	X	PCIE_MSI
2	X	PCIE_INT
1	X	USB3
31:0	X	FIQ31_FIQ0: Flags set by Hardware, cleared by SW

Bit	Reset	Description
0	X	SNOR

3.2.5.4 QUAD_ICTLR_VFIQ_COP_0

FIQ Valid Interrupt Status for COP Register

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

Bit	Reset	Description
23	X	APB_DMA_CH15
22	X	APB_DMA_CH14
21	X	APB_DMA_CH13
20	X	APB_DMA_CH12
19	X	APB_DMA_CH11
18	X	APB_DMA_CH10
17	X	APB_DMA_CH9
16	X	APB_DMA_CH8
15	X	APB_DMA_CH7
14	X	APB_DMA_CH6
13	X	APB_DMA_CH5
12	X	APB_DMA_CH4
11	X	APB_DMA_CH3
10	X	APB_DMA_CH2
9	X	APB_DMA_CH1
8	X	APB_DMA_CH0
4	X	PCIE_WAKE
3	X	PCIE_MSI
2	X	PCIE_INT
1	X	USB3
31:0	X	FIQ31_FIQ0: Flags set by Hardware, cleared by SW
0	X	SNOR

3.2.5.5 QUAD_ICTLR_ISR_0

Latched Interrupt Status Register (HW)

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

Bit	Reset	Description
23	X	APB_DMA_CH15

Bit	Reset	Description
22	X	APB_DMA_CH14
21	X	APB_DMA_CH13
20	X	APB_DMA_CH12
19	X	APB_DMA_CH11
18	X	APB_DMA_CH10
17	X	APB_DMA_CH9
16	X	APB_DMA_CH8
15	X	APB_DMA_CH7
14	X	APB_DMA_CH6
13	X	APB_DMA_CH5
12	X	APB_DMA_CH4
11	X	APB_DMA_CH3
10	X	APB_DMA_CH2
9	X	APB_DMA_CH1
8	X	APB_DMA_CH0
4	X	PCIE_WAKE
3	X	PCIE_MSI
2	X	PCIE_INT
1	X	USB3
31:0	X	ISR31_ISR0: Read-only. Set by hardware event, cleared at source by software.
0	X	SNOR

3.2.5.6 QUAD_ICTLR_FIR_0

Forced Interrupt Status Register (SW)

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

Bit	Reset	Description
23	X	APB_DMA_CH15
22	X	APB_DMA_CH14
21	X	APB_DMA_CH13
20	X	APB_DMA_CH12
19	X	APB_DMA_CH11
18	X	APB_DMA_CH10
17	X	APB_DMA_CH9
16	X	APB_DMA_CH8

Bit	Reset	Description
15	X	APB_DMA_CH7
14	X	APB_DMA_CH6
13	X	APB_DMA_CH5
12	X	APB_DMA_CH4
11	X	APB_DMA_CH3
10	X	APB_DMA_CH2
9	X	APB_DMA_CH1
8	X	APB_DMA_CH0
4	X	PCIE_WAKE
3	X	PCIE_MSI
2	X	PCIE_INT
1	X	USB3
31:0	X	FIR31_FIR0: Read only: Set during write to FIR_SET, cleared during write to FIR_CLR.
0	X	SNOR

3.2.5.7 QUAD_ICTLR_FIR_SET_0

Force Interrupt Register Set

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

Bit	Reset	Description
23	0x0	APB_DMA_CH15
22	0x0	APB_DMA_CH14
21	0x0	APB_DMA_CH13
20	0x0	APB_DMA_CH12
19	0x0	APB_DMA_CH11
18	0x0	APB_DMA_CH10
17	0x0	APB_DMA_CH9
16	0x0	APB_DMA_CH8
15	0x0	APB_DMA_CH7
14	0x0	APB_DMA_CH6
13	0x0	APB_DMA_CH5
12	0x0	APB_DMA_CH4
11	0x0	APB_DMA_CH3
10	0x0	APB_DMA_CH2

Bit	Reset	Description
9	0x0	APB_DMA_CH1
8	0x0	APB_DMA_CH0
4	0x0	PCIE_WAKE
3	0x0	PCIE_MSI
2	0x0	PCIE_INT
1	0x0	USB3
31:0	0x0	FIR_SET: Set Forced Interrupt Bit. Writing a 1 will set an interrupt
0	0x0	SNOR

3.2.5.8 QUAD_ICTLR_FIR_CLR_0

Force Interrupt Register Clear Register

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

Bit	Reset	Description
23	0x0	APB_DMA_CH15
22	0x0	APB_DMA_CH14
21	0x0	APB_DMA_CH13
20	0x0	APB_DMA_CH12
19	0x0	APB_DMA_CH11
18	0x0	APB_DMA_CH10
17	0x0	APB_DMA_CH9
16	0x0	APB_DMA_CH8
15	0x0	APB_DMA_CH7
14	0x0	APB_DMA_CH6
13	0x0	APB_DMA_CH5
12	0x0	APB_DMA_CH4
11	0x0	APB_DMA_CH3
10	0x0	APB_DMA_CH2
9	0x0	APB_DMA_CH1
8	0x0	APB_DMA_CH0
4	0x0	PCIE_WAKE
3	0x0	PCIE_MSI
2	0x0	PCIE_INT
1	0x0	USB3
31:0	0x0	FIR_CLR: Clear Forced Interrupt Bit: Writing a 1 will clear the forced interrupt

Bit	Reset	Description
0	0x0	SNOR

3.2.5.9 QUAD_ICTLR_CPU_IER_0

Enabled Interrupt Source for CPU Register

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

Bit	Reset	Description
23	X	APB_DMA_CH15
22	X	APB_DMA_CH14
21	X	APB_DMA_CH13
20	X	APB_DMA_CH12
19	X	APB_DMA_CH11
18	X	APB_DMA_CH10
17	X	APB_DMA_CH9
16	X	APB_DMA_CH8
15	X	APB_DMA_CH7
14	X	APB_DMA_CH6
13	X	APB_DMA_CH5
12	X	APB_DMA_CH4
11	X	APB_DMA_CH3
10	X	APB_DMA_CH2
9	X	APB_DMA_CH1
8	X	APB_DMA_CH0
4	X	PCIE_WAKE
3	X	PCIE_MSI
2	X	PCIE_INT
1	X	USB3
31:0	X	IER31_IER0: Interrupt Enable Status. 0 = Disabled
0	X	SNOR

3.2.5.10 QUAD_ICTLR_CPU_IER_SET_0

Set Interrupt Enable for CPU Register

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

Bit	Reset	Description
23	0x0	APB_DMA_CH15

Bit	Reset	Description
22	0x0	APB_DMA_CH14
21	0x0	APB_DMA_CH13
20	0x0	APB_DMA_CH12
19	0x0	APB_DMA_CH11
18	0x0	APB_DMA_CH10
17	0x0	APB_DMA_CH9
16	0x0	APB_DMA_CH8
15	0x0	APB_DMA_CH7
14	0x0	APB_DMA_CH6
13	0x0	APB_DMA_CH5
12	0x0	APB_DMA_CH4
11	0x0	APB_DMA_CH3
10	0x0	APB_DMA_CH2
9	0x0	APB_DMA_CH1
8	0x0	APB_DMA_CH0
4	0x0	PCIE_WAKE
3	0x0	PCIE_MSI
2	0x0	PCIE_INT
1	0x0	USB3
31:0	0x0	CPU_IER_SET: Writing a 1 in any bit position will enable the corresponding Interrupt Source for CPU
0	0x0	SNOR

3.2.5.11 QUAD_ICTLR_CPU_IER_CLR_0

CPUs Clear Interrupt Enable for CPU

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

Bit	Reset	Description
23	0x0	APB_DMA_CH15
22	0x0	APB_DMA_CH14
21	0x0	APB_DMA_CH13
20	0x0	APB_DMA_CH12
19	0x0	APB_DMA_CH11
18	0x0	APB_DMA_CH10
17	0x0	APB_DMA_CH9

Bit	Reset	Description
16	0x0	APB_DMA_CH8
15	0x0	APB_DMA_CH7
14	0x0	APB_DMA_CH6
13	0x0	APB_DMA_CH5
12	0x0	APB_DMA_CH4
11	0x0	APB_DMA_CH3
10	0x0	APB_DMA_CH2
9	0x0	APB_DMA_CH1
8	0x0	APB_DMA_CH0
4	0x0	PCIE_WAKE
3	0x0	PCIE_MSI
2	0x0	PCIE_INT
1	0x0	USB3
31:0	0x0	CPU_IER_CLR: Writing a 1 in any bit position will disable the corresponding Interrupt Source for CPU
0	0x0	SNOR

3.2.5.12 QUAD_ICTLR_CPU_IEP_CLASS_0

CPUs Interrupt Enable Priority Class (FIQ/IRQ)

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

Bit	Reset	Description
23	0x0	APB_DMA_CH15
22	0x0	APB_DMA_CH14
21	0x0	APB_DMA_CH13
20	0x0	APB_DMA_CH12
19	0x0	APB_DMA_CH11
18	0x0	APB_DMA_CH10
17	0x0	APB_DMA_CH9
16	0x0	APB_DMA_CH8
15	0x0	APB_DMA_CH7
14	0x0	APB_DMA_CH6
13	0x0	APB_DMA_CH5
12	0x0	APB_DMA_CH4
11	0x0	APB_DMA_CH3

Bit	Reset	Description
10	0x0	APB_DMA_CH2
9	0x0	APB_DMA_CH1
8	0x0	APB_DMA_CH0
4	0x0	PCIE_WAKE
3	0x0	PCIE_MSI
2	0x0	PCIE_INT
1	0x0	USB3
31:0	0x0	CPU_IEP_CLASS: Set Priority Interrupt Source For CPU. 1 = FIQ, 0 = IRQ.
0	0x0	SNOR

3.2.5.13 QUAD_ICTLR_COP_IER_0

Enabled Interrupt Source for COP Register

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

Bit	Reset	Description
23	X	APB_DMA_CH15
22	X	APB_DMA_CH14
21	X	APB_DMA_CH13
20	X	APB_DMA_CH12
19	X	APB_DMA_CH11
18	X	APB_DMA_CH10
17	X	APB_DMA_CH9
16	X	APB_DMA_CH8
15	X	APB_DMA_CH7
14	X	APB_DMA_CH6
13	X	APB_DMA_CH5
12	X	APB_DMA_CH4
11	X	APB_DMA_CH3
10	X	APB_DMA_CH2
9	X	APB_DMA_CH1
8	X	APB_DMA_CH0
4	X	PCIE_WAKE
3	X	PCIE_MSI
2	X	PCIE_INT
1	X	USB3

Bit	Reset	Description
31:0	X	IER31_IER0: Interrupt Enable Status. 0 = Disabled.
0	X	SNOR

3.2.5.14 QUAD_ICTLR_COP_IER_SET_0

Set Interrupt Source for COP Register

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

Bit	Reset	Description
23	0x0	APB_DMA_CH15
22	0x0	APB_DMA_CH14
21	0x0	APB_DMA_CH13
20	0x0	APB_DMA_CH12
19	0x0	APB_DMA_CH11
18	0x0	APB_DMA_CH10
17	0x0	APB_DMA_CH9
16	0x0	APB_DMA_CH8
15	0x0	APB_DMA_CH7
14	0x0	APB_DMA_CH6
13	0x0	APB_DMA_CH5
12	0x0	APB_DMA_CH4
11	0x0	APB_DMA_CH3
10	0x0	APB_DMA_CH2
9	0x0	APB_DMA_CH1
8	0x0	APB_DMA_CH0
4	0x0	PCIE_WAKE
3	0x0	PCIE_MSI
2	0x0	PCIE_INT
1	0x0	USB3
31:0	0x0	COP_IER_SET: Writing a 1 in any bit position will enable the corresponding Interrupt Source for COP
0	0x0	SNOR

3.2.5.15 QUAD_ICTLR_COP_IER_CLR_0

Clear Interrupt Source for COP Register

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

Bit	Reset	Description
23	0x0	APB_DMA_CH15
22	0x0	APB_DMA_CH14
21	0x0	APB_DMA_CH13
20	0x0	APB_DMA_CH12
19	0x0	APB_DMA_CH11
18	0x0	APB_DMA_CH10
17	0x0	APB_DMA_CH9
16	0x0	APB_DMA_CH8
15	0x0	APB_DMA_CH7
14	0x0	APB_DMA_CH6
13	0x0	APB_DMA_CH5
12	0x0	APB_DMA_CH4
11	0x0	APB_DMA_CH3
10	0x0	APB_DMA_CH2
9	0x0	APB_DMA_CH1
8	0x0	APB_DMA_CH0
4	0x0	PCIE_WAKE
3	0x0	PCIE_MSI
2	0x0	PCIE_INT
1	0x0	USB3
31:0	0x0	COP_IER_CLR: Writing a 1 in any bit position will disable the corresponding Interrupt Source for COP
0	0x0	SNOR

3.2.5.16 QUAD_ICTLR_COP_IEP_CLASS_0

COPs Interrupt Enable Priority Class (FIQ/IRQ) Register

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

Bit	Reset	Description
23	0x0	APB_DMA_CH15
22	0x0	APB_DMA_CH14
21	0x0	APB_DMA_CH13

Bit	Reset	Description
20	0x0	APB_DMA_CH12
19	0x0	APB_DMA_CH11
18	0x0	APB_DMA_CH10
17	0x0	APB_DMA_CH9
16	0x0	APB_DMA_CH8
15	0x0	APB_DMA_CH7
14	0x0	APB_DMA_CH6
13	0x0	APB_DMA_CH5
12	0x0	APB_DMA_CH4
11	0x0	APB_DMA_CH3
10	0x0	APB_DMA_CH2
9	0x0	APB_DMA_CH1
8	0x0	APB_DMA_CH0
4	0x0	PCIE_WAKE
3	0x0	PCIE_MSI
2	0x0	PCIE_INT
1	0x0	USB3
31:0	0x0	COP_IEP_CLASS: Set Priority Interrupt Source For COP. 1 = FIQ, 0 = IRQ.
0	0x0	SNOR

4.0 ARBITRATION SEMAPHORES

4.1 Overview

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

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

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

4.2 Semaphore Registers

4.2.1 ARB_SEMA_SMP_GNT_ST_0

Semaphore Granted Status Register

Offset: 000h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

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

4.2.2 ARB_SEMA_SMP_GET_0

Request Arbitration Semaphore Register

Offset: 004h | Read/Write: WO | Reset: 0b00000000000000000000000000000000

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

4.2.3 ARB_SEMA_SMP_PUT_0

Arbitration Semaphore Put Request Register

Offset: 008h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

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

4.2.4 ARB_SEMA_SMP_REQ_ST_0

Arbitration Request Pending Status (1=PENDING) Register

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

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

5.0 CLOCK AND RESET CONTROLLER

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

CLKGEN provides the registers to program the PLLs and control most of the clock source programming and most of the clock dividers. CLKGEN input signals include the external clock for the reference frequency (12 MHz, 13 MHz, 19.2 MHz, or 26 MHz) and the external clock for the Real Time Clock (32.768 kHz). The outputs from CLKGEN are mostly clocks to the other blocks in the Tegra® 2 Processor system.

RSTGEN provides the registers needed to control resetting each block in the Tegra 2 Series.

5.1 Hardware Features

5.1.1 External Clock Sources

The CLKGEN block takes two external clock sources as input:

- A single external 32.768 kHz clock, normally provided by the Power Management Unit (PMU)
- A single crystal oscillator at one of these frequencies: 12 MHz, 13 MHz, 19.2 MHz, or 26 MHz. 12 MHz is for Personal Media Players (PMP) where it is expected to be provided by external crystal, or where the Tegra 2 Series devices have a dedicated crystal. Support for 13, 19.2, or 26 MHz is intended for cell phones where it is expected to be provided by the Baseband Processor. Lower frequencies are generally lower power.

5.1.2 PLLs

There are twelve PLLs in the Tegra 2 Series clock system:

- PLLX is dedicated to the Cortex-A9 CPU complex, and provides a high speed CPU clock when required. It operates up to 1.1 GHz or higher.
- PLLC is a general purpose PLL, whose nominal maximum frequency is 600 MHz. This PLL is available as a clock source to many modules.
- PLLP is a "fixed" 216 MHz source for most of the Peripherals on the Tegra 2 Series devices. It should always be set to this frequency. It is available as a source to most modules. In addition, PLLP is the source of a number of standard divided-down frequencies that are also available as clock sources to other modules.
- PLLM is the source for the External Memory Controller (EMC) 2x clock. It is dedicated to this function. The nominal maximum frequency required by EMC on the Tegra 2 Series devices is 800 MHz. PLLM is available as a source to other modules as well.
- PLLA is the source for Audio, and is used to generate exact 11.2896, 12.288 or 24.576 MHz for audio codecs and audio-related devices.
- PLLD is the source for DSI and for the Display subsystem in general. It provides the frequencies required by DSI and the Display controller when DSI is in use. It is also a dedicated Display PLL to provide the frequencies required by Display, HDMI, TVOUT when a precise enough frequency cannot be provided by one of the other Tegra 2 Series device PLLs. Wherever possible, the Display subsystem should use a clock source other than PLLD, since PLLD consumes substantially higher power than the other (non-MIPI) PLLs in the system.
- PLLU is the source for the USB PHY and controllers. It is dedicated to provide the 12 MHz required by the USB PHY, and 60 MHz and 480 MHz USB clocks.
- PLLE (Supported only on Tegra® 250) is dedicated to the PCIE interface, and generates the required 100 MHz reference clock used for the PCIE PHY.
- PLLS can be used in some low power scenarios to generate a 12/13/19.2/26 MHz clock from the 32.768 kHz reference clk. It is higher jitter than using the crystal reference, but potentially lower power.

- Three additional PLLs are embedded in their respective interface controllers. The USB controller has an internal PLL, as does the TMDS HDMI PHY and the PCIE PHY. These PLLs are programmed via their respective controllers' registers, and the clocks generated by these PLLs are neither visible to CLKGEN nor to the rest of the system.

5.1.3 Clock Dividers / Skippers / Multipliers

There are a number of clock-skippers, clock-dividers and clock-multipliers in use in the Tegra 2 Series processors.

- An M/N clock-skipping divider that is used to generate the CPU clock and also the Tegra 2 Series "system clock" (sclk).
- Clock skippers that are used to generate the "AHB clock" (hclk) and "APB clock" (pclk). One that divides down by a 2b unsigned value (U2).
- An unsigned 16b divider (U16) that is the divider for the I2C and UART devices.
- A times-two multiplier used to double the external reference frequency, and used to double the audio clock.
- An unsigned 8b divider that provides 7b of mantissa and 1b of fraction (U7.1). Tegra 2 has reduced the number of divider types and in particular this U7.1 divider is the default divider for most all blocks in Tegra 2 Series processors.

5.1.4 Clock Sources

The clock sources are described as follows:

- PLLM: Clock source for EMC 2x clock
- PLLX: Clock source for the CPU
- PLLC: Clock source for general use
- PLLP: Clock source for most peripherals
- PLLA: Audio clock sources:
  - 11.2896 MHz
  - 12.288 MHz
  - 24.576 MHz
- PLLU: Clock source for USB PHY, provides 12/60/480 MHz
- PLLD: Clock source for the display subsystem
- OSC: Source from external crystal
- CLK_32K_IN: 32 kHz clock provided by the PMC

Almost every device or block in the Tegra 2 Series can be driven from a selection of clock sources. There are three general types of source selection:

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

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

The clocks available as sources in Tegra 2 are shown in Table 9.

Table 9. Primary Clock Sources in the Tegra 2 Series devices

Clock name	Description
osc	This clock runs at 12 MHz, 13 MHz, 19.2 MHz, or 26 MHz. It is 12 MHz if PCIe (Tegra 250 only) is used.
clk_m	Alternate name for osc.
ck32khz_IB	This clock runs at 32.768 kHz.
clk_s	Alternate name for ck32khz_IB.
sclk	This is the system clock which can run at max 200MHz.
pllP_out	This is the PLLP's output clock.
pllC_out	This is the PLLC's output clock.
pllM_out	This is the PLLM's output clock.
int_pllA_out	This is the PLLA's output clock.
pllD_out0	This is the divided-by-2 (max @ 500MHz) from PLLD's output clock.
pllX_out	This is the PLLX's output clock.
pllS_out	This is the PLLS output clock. Its purpose is to replace the osc clock coming in to the chip for some application by multiplying the 32.768 kHz reference clock up to create the 12 MHz, 13 MHz, 19.2 MHz, or 26 MHz

Table 10. Derived Clock Sources in the Tegra 2 Series devices

Clock name	Description
clk_d:	This is the clock doubled from "osc".
pllC_out1:	This is the U7.1 NV divider output clocked by "pllC_out".
pllP_out1:	This is the U7.1 NV divider output (fix @ 28.8MHz) clocked by "pllP_out".
pllP_out2:	This is the U7.1 NV divider output (fix @ 48MHz) clocked by "pllP_out".
pllP_out3:	This is the U7.1 NV divider output (fix @ 72MHz) clocked by "pllP_out".
pllP_out4:	This is the U7.1 NV divider output (fix @ (108MHz) clocked by "pllP_out".
pllM_out1:	This is the U7.1 NV divider output clocked by "pllM_out".
pllA_out0:	This is the U7.1 NV divider output clocked by "int_pllA_out".
audio_sync_clk:	This clock is muxed from 8 possible audio clock inputs.
audio_2x_sync_clk:	This is the clock doubled from "audio_sync_clk".

Table 11 Clock Source Muxes

	osc	pllC_out	ck32khz_IB	pllM_out	pllP_out	pllP_out4	pllP_out3	clk_d	pllX_out	pllC_out1	pllP_out2	pllM_out1	pllA_out0	audio_2x_sync_clk	pllD_out0	int_pllA_out	Have super clock divider ? (# of bits)	Have NV divider? (# of bits)
CPU	0	1	2	3	4	5	6	7	8								8	
SYS/COP	0		6			2	3	5		1	4	7					8	
I2S[1:2]	3				2								0	1				8
SPDIF_OUT	3				2								0	1				8
SPDIF_IN		1		2	0													8
PWM	3	1	4		0									2				8
SPI_CTLR	3	1		2	0													8

	osc	pllC_out	ck32khz_lB	pllM_out	pllP_out	pllP_out4	pllP_out3	clk_d	pllX_out	pllC_out1	pllP_out2	pllM_out1	pllA_out0	audio_2x_sync_clk	pllD_out0	int_pllA_out	Have super clock divider ? (# of bits)	Have NV divider? (# of bits)
XIO	3	1		2	0													8
TWC	3	1		2	0													8
SBC[1:4] (SPI 1:4)	3	1		2	0													8
IDE	3	1		2	0													8
NDFLASH	3	1		2	0													8
VFIR	3	1		2	0													8
SDMMC[1:4]	3	1		2	0													8
VDE	3	1		2	0													8
CSITE	3	1		2	0													8
LA	3	1		2	0													8
OWR	3	1		2	0													8
NOR	3	1		2	0													8
MIPI	3	1		2	0													8
I2C[1:3]	3	1		2	0													16
DVC	3	1		2	0													16
UART[1:5]	3	1		2	0													16
3D		1		0	2								3					8
2D		1		0	2								3					8
VI		1		0	2								3					8
VI_SENSOR		1		0	2								3					8
EPP		1		0	2								3					8
MPE		1		0	2								3					8
HOST1X		1		0	2								3					8
CVE	3	2			0										1			8
TVO	3	2			0										1			8
HDMI	3	2			0										1			8
TVDAC	3	2			0										1			8
DISP[1:2]	3	2			0										1			8
EMC (2x/1x)	3	1		0/4[a]	2													8
pllC_out1		x																8
pllM_out1				x														8
pllP_out1					x													8
pllP_out2					x													8
pllP_out3					x													8
pllP_out4					x													8
pllA_out0																x		8

Note: [a] When EMC clock source = 4, the NV divider will be bypassed/ignored. This setting will give a very short low jitter clock path from pllM_out to EMC clock.

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

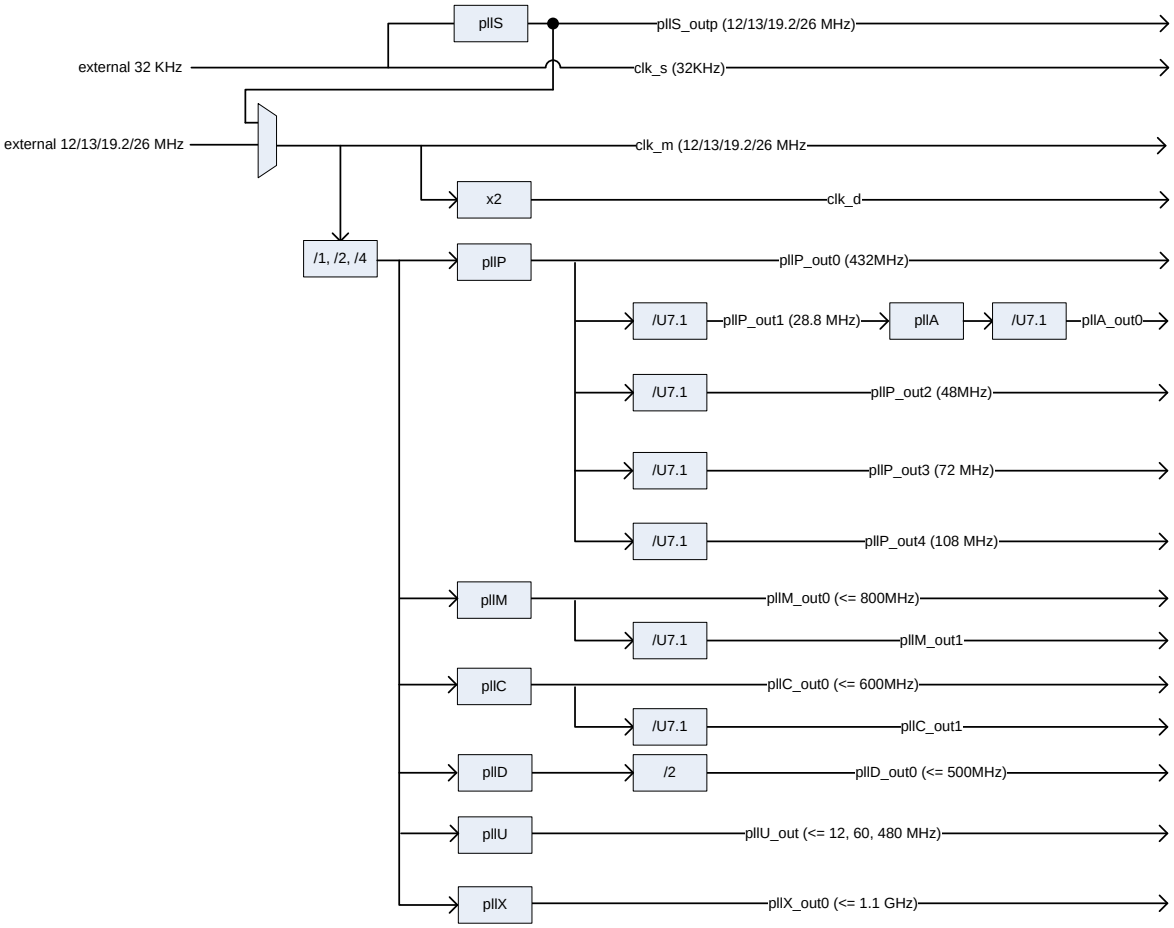
Apart from the programmable clock sources each peripheral has, there are a number of modules which take in an additional fix PLL clock source for a portion of their logic. The table below indicates which fix PLL output is used by which peripheral.

Table 12. PLL Clock Usage

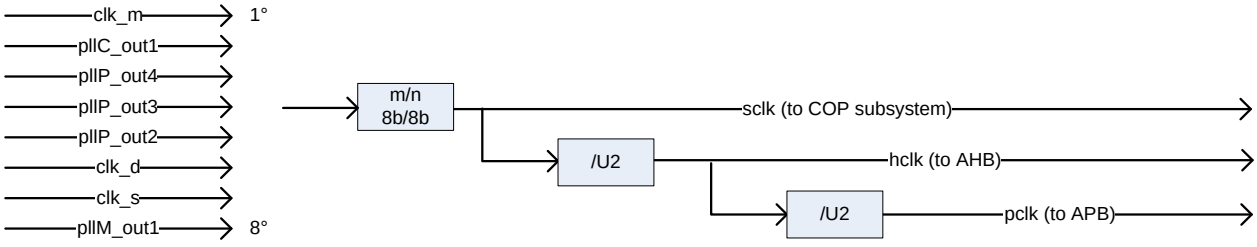
Module/Logic Fix	PLL clock source used
LFSR	PLLM_out0
DSI	PLLP_out3
CSI	PLLP_out3
I2C1	PLLP_out3
I2C2	PLLP_out3
I2C3	PLLP_out3
UART1	PLLP_out3
UART2	PLLP_out3
UART3	PLLP_out3
UART4	PLLP_out3
UART5	PLLP_out3

5.2 Clocking Block Diagrams

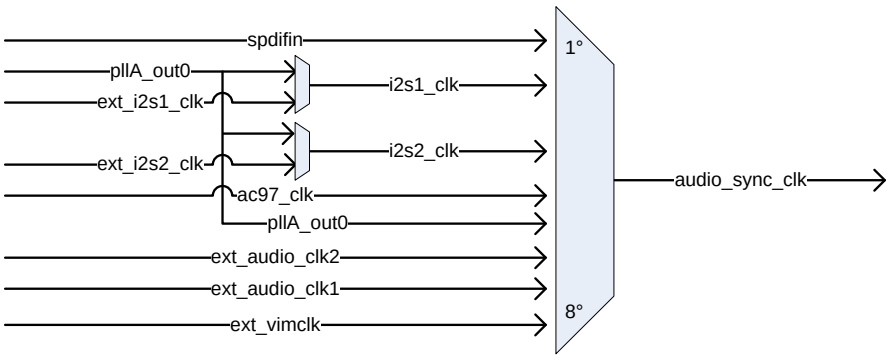
5.2.1 Clock Sources

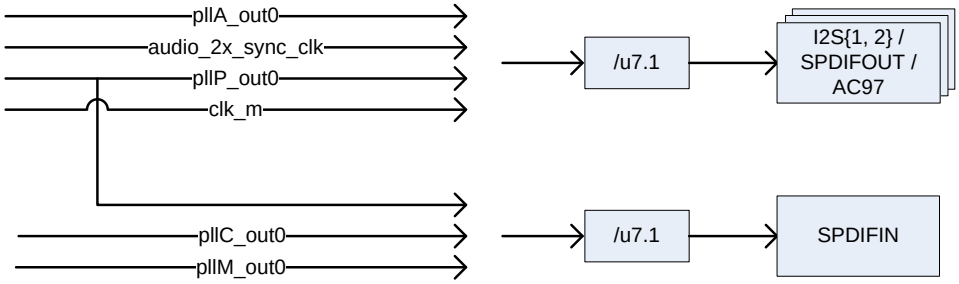


5.2.2 Derived System Clocks

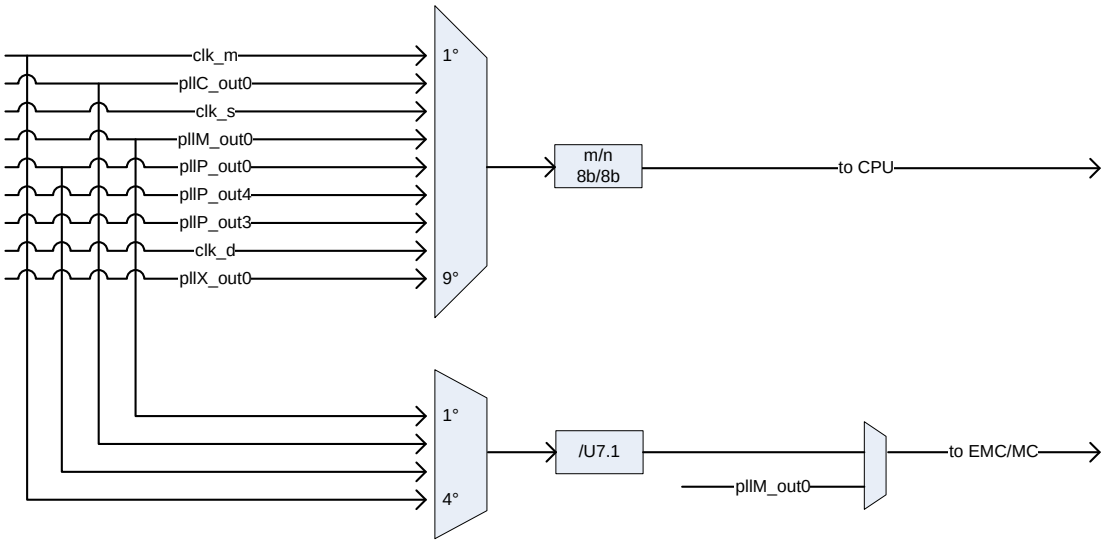


5.2.3 Audio Clocks

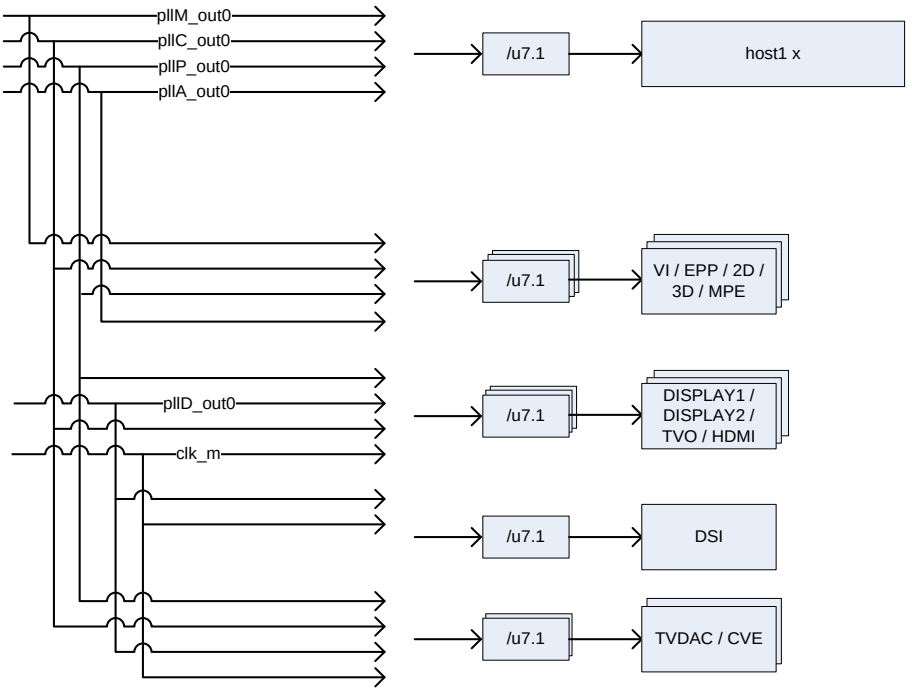




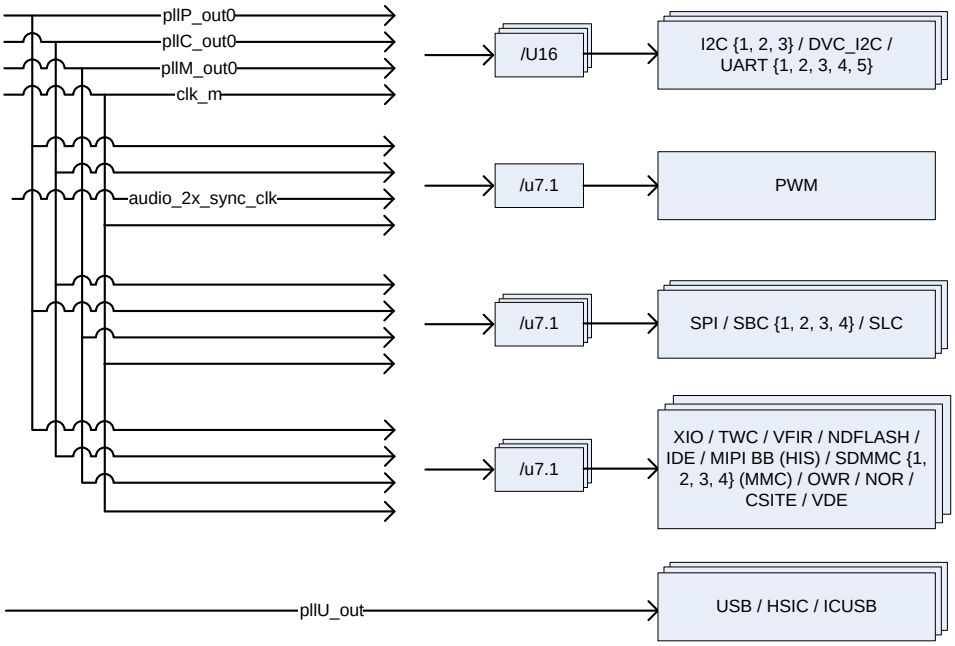
5.2.4 CPU Clocks



5.2.5 Graphics Clocks



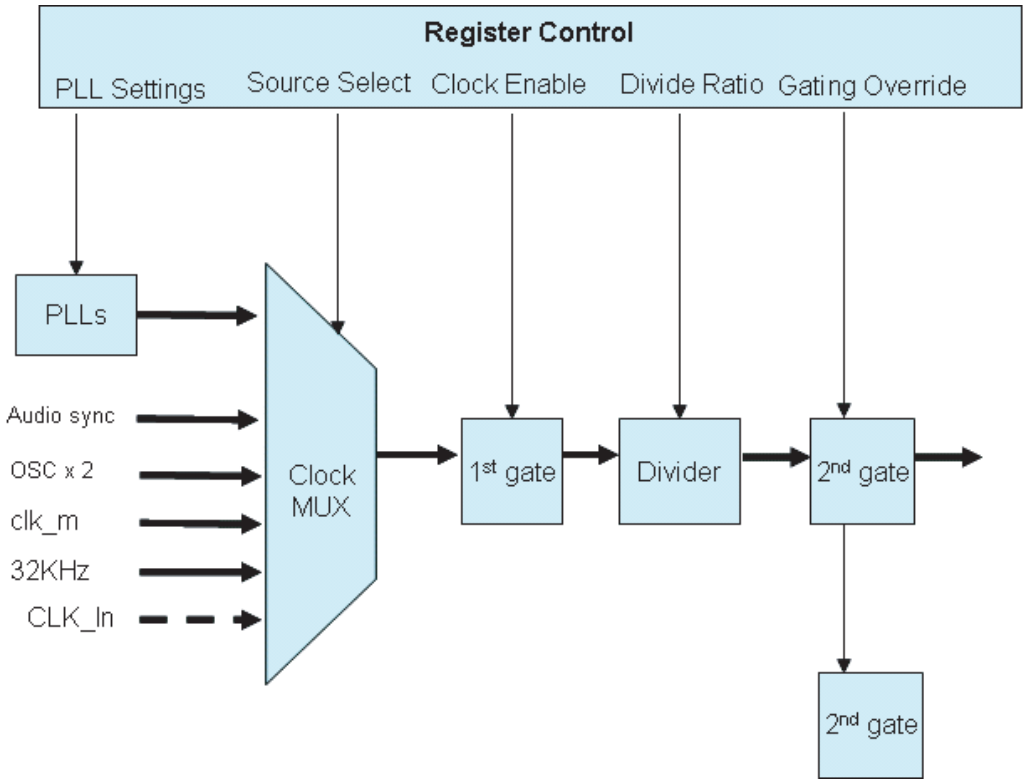
5.2.6 Device Clocks



5.3 Software Features and Programming Model

5.3.1 Clock Control

The figure below illustrates the software clock control model.



5.3.2 PLL Programming

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

VCO = (Fi / M) * N

Fo = VCO / (2 ^ P)

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

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

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

To change a PLL, do the following:

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

5.3.3 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)

5.3.4 Changing Clock Sources and Clock Dividers

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

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

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

Sequence 1 to change clock source and divider:

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

Sequence 2 to change clock source and divider:

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

5.3.5 Clock Gating (Override)

Please refer to the register definition section for more information.

5.3.5.1 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.

5.3.6 Use-Case Restrictions

5.3.6.1 Use of PLLC for Display

Other than PLLD, there is no dedicated PLL for either Display controller. PLLD is required when one of the Display controllers outputs to DSI, but should be otherwise avoided due to its higher power. When DSI is not one of the output devices, or in dual-display use cases, the additional PLLs PLLC and PLLP must be used. The matrix of choices is shown in the Table below.

Table 13 PLL sources for Display based on use-case

Primary Head	PLL Choices	Secondary Head	PLL Choices
Parallel	P, C, D	--	--
DSI	D	--	--
Parallel	P, C	Sub	P
DSI	D	Sub	P
Parallel	P, C	DSI	D
Parallel	P, C, D	Parallel	P, C, D
Parallel	P, C, D	TVO	P
Parallel	P, C, D	HDMI	P (480p), C, D (720p)
Parallel	P, C, D	CRT	C, D
DSI	D	TVO	P
DSI	D	HDMI	P (480p), C, D (720p)
DSI	D	CRT	C, D

If possible, the first choice should use the fixed PLLP as Display source clock (divided-down to the required frequency). However, many Display resolutions and/or output devices cannot be driven within the required tolerance by a divided-down PLLP.

Second choice should be to use either PLLD or PLLC, depending on the trade-off that is acceptable. If PLLD is used, power consumption is 5x higher than PLLC. If PLLC is used, it may compromise some other use cases since the Display source PLL must be programmed to a precise frequency to achieve the required Display error tolerance.

Since PLLC is a possible source clock for the AVP, the high-speed serial devices (SFLASH and SPI), for the video encoder (MPE), and for 2D and 3D, these devices may not achieve their target (maximum) frequencies when Display must constrain PLLC directly.

An example of this is shown in the Table below for a case where external memory is 166 MHz DDR (EMC 2x clock = 333 MHz).

Table 14. Example of device frequencies possible when PLLC frequency is constrained for use by Display

PLLC	PLLP	AVP	HS serial	MPE	2D / 3D
600.0	216.0	150.0	200.0	200.0	216.0
590.0	216.0	147.5	196.7	196.7	216.0
580.0	216.0	145.0	193.3	193.3	216.0
570.0	216.0	144.0	190.0	190.0	216.0
560.0	216.0	144.0	186.7	186.7	216.0
550.0	216.0	144.0	183.3	183.3	275.0
540.0	216.0	144.0	180.0	180.0	270.0
530.0	216.0	144.0	176.7	176.7	265.0
520.0	216.0	144.0	173.3	173.3	260.0
510.0	216.0	144.0	170.0	170.0	255.0
500.0	216.0	144.0	166.7	166.7	250.0

5.4 Clock and Reset Controller Registers

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

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

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

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

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

- CoreSight

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 eliminate the chance of either processors trying to talk to a boot device while it's still in reset state.

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

Precaution

- All modules support changing the clock divider ratio without disabling the clock.
- All modules' clock switching is glitch free except "audio_sync_clk".
- For "audio_sync_clk", the clock source select needs to be setup ahead of time before changing the device clock source to use that audio clock or to enable the device which use that audio clock.
- Changing a module's clock source and clock divider value at the same time is not allowed. Hardware cannot guarantee which one is going to change first and thus, can create a potential problem where the intermediate frequency going to a device may be higher than what's tolerable by that device.
- Before stopping clock (via CLK_OUT_ENB_L/H/U registers) and/or asserting reset (via RST_DEVICES_L/H/U registers) to a module, it's very important to first check the module to make sure it's not active. Stopping clock/asserting reset while the module is still busy can cause relatively minor problem such as incorrect data read/written, or catastrophic problem such as system hang. To ensure a module is not active, (a) disable the module by programming its disable bit if not already done so, and (b) wait until the module is not active by checking for its busy bit, done bit, count, or similar mechanism.

To setup a non-boot device for operation (only apply if a device has a CLK_SOURCE_ register)

1. Make sure the device's reset is asserted (via RST_DEVICES_L/H/U registers).
2. Enable clock to the device (via CLK_OUT_ENB_L/H/U registers).
3. Change the clock divisor to the device (via CLK_SOURCE_ register).
4. Wait 1us to make sure clock divider has changed.
5. Change the clock source to the device (via CLK_SOURCE_ register).
6. Wait 2us to make sure clock source/device logic is stabilized.
7. Deassert device's reset (via RST_DEVICES_L/H/U registers).

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

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

1. Make sure the device is disabled (via the device's register).
2. Assert device's reset (via RST_DEVICES_L/H/U registers).
3. Make sure clock to the device is enabled (via CLK_OUT_ENB_L/H/U registers).
4. Change the clock divisor to the device (via CLK_SOURCE_ register).
5. Wait 1us to make sure clock divider has changed.
6. Change the clock source to the device (via CLK_SOURCE_ register).
7. Wait 2us to make sure clock source/device logic is stabilized.

8. Deassert device's reset (via RST_DEVICES_L/H/U registers).

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

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

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

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

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

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

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

Main clock sources used by the system:

(A) Primary clocks.

- "osc" or "clk_m" which can be either 12MHz, 13MHz, 19.2MHz, or 26Mhz.
- "clk_s" which is 32 kHz.

(B) Derived clocks.

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

(C) PLL clocks.

- "PLLC" which is general purpose and its output is called "pllC_out0".
- "PLLM"(Memory) which is general purpose and its output is called "pllM_out0".
- "PLLP"(Peripheral) which is fixed at 432MHz and its output is called "pllP_out0".
- "PLLA"(Audio) which is cascaded from PLLP and is use for audio purposes.
- "PLLU"(USB) which have 3 outputs and fixed at 12MHz, 60MHz, and 480MHz.
- "PLLD"(DSI) which can go up-to 1GHz and its output (after a fix /2) is called "pllD_out0".
- "PLLX" which is extra high frequency used only by Cortex and is called "pllX_out0".
- "PLLE" which is only use by PCIE (PLLE is supported only on Tegra 250).
- "PLLS" (clocked by 32.768 kHz) which is used to generate replacement to osc (when osc pad is turn off).

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

- "pllC_out1" is divided-down from "pllC_out0"
- "pllM_out1" is divided-down from "pllM_out0"

- "pllP_out1" is divided-down from "pllP_out0" and is fixed at 28.8 MHz
- "pllP_out2" is divided-down from "pllP_out0" and is fixed at 48.0 MHz
- "pllP_out3" is divided-down from "pllP_out0" and is fixed at 72.0 MHz
- "pllP_out4" is divided-down from "pllP_out0" and is fixed at 108. 0MHz
- "pllA_out0" is divided-down from "pllA" output

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

5.4.1 CLK_RST_CONTROLLER_RST_SOURCE_0

Offset: 000h | Read/Write: R/W | Reset: 0bxxxxxxxx00x000

Bit	R/W	Reset	Description
13	RO	X	SWR_SYS_RST_STA: System reset by SW (RO)
12	RO	X	WDT_SYS_RST_STA: System reset by watch dog timer (RO)
11	RO	X	SWR_COP_RST_STA: COP reset by SW (RO)
10	RO	X	WDT_COP_RST_STA: COP reset by watch dog timer (RO)
9	RO	X	SWR_CPU_RST_STA: CPU reset by SW (RO)
8	RO	X	WDT_CPU_RST_STA: CPU reset by watch dog timer (RO)
5	RW	DISABLE	WDT_EN: Enable Watch Dog Timer (Dead Man Timer) 0 = DISABLE 1 = ENABLE
4	RW	0x0	WDT_SEL: Watch Dog Timer Select
2	RW	0x0	WDT_SYS_RST_EN: Enable Watch Dog Timer reset for system.
1	RW	0x0	WDT_COP_RST_EN: Enable Watch Dog Timer reset for COP
0	RW	0x0	WDT_CPU_RST_EN: Enable Watch Dog Timer reset for CPU

5.4.2 CLK_RST_CONTROLLER_RST_DEVICES_L_0

Offset: 004h | Read/Write: R/W | Reset: 0b0x11111111111111111111011001001

Bit	Reset	Description
31	DISABLE	SWR_CACHE2_RST: Reset COP cache controller. 0 = DISABLE 1 = ENABLE
29	ENABLE	SWR_VCP_RST: Reset vector co-processor. 0 = DISABLE 1 = ENABLE
28	ENABLE	SWR_HOST1X_RST: Reset HOST1X. 0 = DISABLE 1 = ENABLE
27	ENABLE	SWR_DISP1_RST: Reset DISP1 controller. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
26	ENABLE	SWR_DISP2_RST: Reset DISP2 controller. 0 = DISABLE 1 = ENABLE
25	ENABLE	SWR_IDE_RST: Reset IDE controller. 0 = DISABLE 1 = ENABLE
24	ENABLE	SWR_3D_RST: Reset 3D controller. 0 = DISABLE 1 = ENABLE
23	ENABLE	SWR_ISP_RST: Reset ISP controller. 0 = DISABLE 1 = ENABLE
22	ENABLE	SWR_USBD_RST: Reset USB controller 0 = DISABLE 1 = ENABLE
21	ENABLE	SWR_2D_RST: Reset 2D graphics engine controller. 0 = DISABLE 1 = ENABLE
20	ENABLE	SWR_VI_RST: Reset VI controller. 0 = DISABLE 1 = ENABLE
19	ENABLE	SWR_EPP_RST: Reset EPP controller. 0 = DISABLE 1 = ENABLE
18	ENABLE	SWR_I2S2_RST: Reset I2S 2 Controller 0 = DISABLE 1 = ENABLE
17	ENABLE	SWR_PWM_RST: Reset Pulse Width Modulator 0 = DISABLE 1 = ENABLE
16	ENABLE	SWR_TWC_RST: Reset Three Wire Controller 0 = DISABLE 1 = ENABLE
15	ENABLE	SWR_SDMMC4_RST: Reset SDMMC4 controller. 0 = DISABLE 1 = ENABLE
14	ENABLE	SWR_SDMMC1_RST: Reset SDMMC1 controller. 0 = DISABLE 1 = ENABLE
13	ENABLE	SWR_NDFLASH_RST: Reset NAND flash controller. 0 = DISABLE 1 = ENABLE
12	ENABLE	SWR_I2C1_RST: Reset I2C1 Controller 0 = DISABLE 1 = ENABLE
11	ENABLE	SWR_I2S1_RST: Reset I2S 1 Controller 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
10	ENABLE	SWR_SPDIF_RST: Reset SPDIF Controller 0 = DISABLE 1 = ENABLE
9	ENABLE	SWR_SDMMC2_RST: Reset SDMMC2 Controller 0 = DISABLE 1 = ENABLE
8	DISABLE	SWR_GPIO_RST: Reset GPIO Controller 0 = DISABLE 1 = ENABLE
7	ENABLE	SWR_UART2_RST: Reset UART2/VFIR Controller 0 = DISABLE 1 = ENABLE
6	ENABLE	SWR_UART1_RST: Reset UART1 Controller 0 = DISABLE 1 = ENABLE
5	DISABLE	SWR_TMR_RST: Reset Timer Controller 0 = DISABLE 1 = ENABLE
3	ENABLE	SWR_AC97_RST: Reset AC97 Controller 0 = DISABLE 1 = ENABLE
2	DISABLE	SWR_TRIG_SYS_RST: Write 1 to pulse System Reset Signal. HW clears this bit 0 = DISABLE 1 = ENABLE
1	DISABLE	SWR_COP_RST: Write 1 to force COP Reset Signal. SW needs to clear this bit when done. 0 = DISABLE 1 = ENABLE
0	ENABLE	SWR_CPU_RST: Write 1 to force CPU Reset Signal. SW needs to clear this bit when done. 0 = DISABLE 1 = ENABLE

5.4.3 CLK_RST_CONTROLLER_RST_DEVICES_H_0

Offset: 008h | Read/Write: R/W | Reset: 0b1111111x11111111111110110111x111

Bit	Reset	Description
31	ENABLE	SWR_BSEV_RST: Reset BSEV controller. 0 = DISABLE 1 = ENABLE
30	ENABLE	SWR_BSEA_RST: Reset BSEA controller. 0 = DISABLE 1 = ENABLE
29	ENABLE	SWR_VDE_RST: Reset VDE & BSEV controller. 0 = DISABLE 1 = ENABLE
28	ENABLE	SWR_MPE_RST: Reset MPE controller. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
27	ENABLE	SWR_USB3_RST: Reset USB3 controller. 0 = DISABLE 1 = ENABLE
26	ENABLE	SWR_USB2_RST: Reset USB2 controller. 0 = DISABLE 1 = ENABLE
25	ENABLE	SWR_EMC_RST: Reset EMC controller. 0 = DISABLE 1 = ENABLE
23	ENABLE	SWR_UART3_RST: Reset UART3 Controller 0 = DISABLE 1 = ENABLE
22	ENABLE	SWR_I2C2_RST: Reset I2C 2 controller. 0 = DISABLE 1 = ENABLE
21	ENABLE	SWR_TVDAC_RST: Reset TVDAC controller. 0 = DISABLE 1 = ENABLE
20	ENABLE	SWR_CSI_RST: Reset CSI controller. 0 = DISABLE 1 = ENABLE
19	ENABLE	SWR_HDMI_RST: Reset HDMI 0 = DISABLE 1 = ENABLE
18	ENABLE	SWR_MIPI_RST: Reset MIPI base-band controller. 0 = DISABLE 1 = ENABLE
17	ENABLE	SWR_TVO_RST: Reset TVO/CVE controller 0 = DISABLE 1 = ENABLE
16	ENABLE	SWR_DSI_RST: Reset DSI controller 0 = DISABLE 1 = ENABLE
15	ENABLE	SWR_DVC_I2C_RST: Reset DVC-I2C Controller 0 = DISABLE 1 = ENABLE
14	ENABLE	SWR_SBC3_RST: Reset SBC 3 (SPI 3) Controller. 0 = DISABLE 1 = ENABLE
13	ENABLE	SWR_XIO_RST: Reset XIO controller. 0 = DISABLE 1 = ENABLE
12	ENABLE	SWR_SBC2_RST: Reset SBC 2 (SPI 2) Controller. 0 = DISABLE 1 = ENABLE
11	ENABLE	SWR_SPI1_RST: Reset SPI 1 Controller 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
10	DISABLE	SWR_SNOR_RST: Reset NOR Flash Controller. 0 = DISABLE 1 = ENABLE
9	ENABLE	SWR_SBC1_RST: Reset SBC 1 (SPI 1) Controller. 0 = DISABLE 1 = ENABLE
8	ENABLE	SWR_KFUSE_RST: Reset KFuse controller. 0 = DISABLE 1 = ENABLE
5	ENABLE	SWR_STAT_MON_RST: Reset statistic monitor 0 = DISABLE 1 = ENABLE
2	ENABLE	SWR_APBDMA_RST: Reset APB-DMA. 0 = DISABLE 1 = ENABLE
1	ENABLE	SWR_AHBDMA_RST: Reset AHB-DMA. 0 = DISABLE 1 = ENABLE
0	ENABLE	SWR_MEM_RST: Reset MC. 0 = DISABLE 1 = ENABLE

5.4.4 CLK_RST_CONTROLLER_RST_DEVICES_U_0

Offset: 00ch | Read/Write: R/W | Reset: 0b010111111111

Bit	Reset	Description
11	DISABLE	SWR_AVPUCQ_RST: Reset AVPUCQ logic. 0 = DISABLE 1 = ENABLE
10	ENABLE	SWR_PCIEXCLK_RST: Reset PCIEXCLK logic. 0 = DISABLE 1 = ENABLE
9	DISABLE	SWR_CSITE_RST: Reset Coresight controller. 0 = DISABLE 1 = ENABLE
8	ENABLE	SWR_AFI_RST: Reset AFI controller. 0 = DISABLE 1 = ENABLE
7	ENABLE	SWR_OWR_RST: Reset OWR controller. 0 = DISABLE 1 = ENABLE
6	ENABLE	SWR_PCIE_RST: Reset PCIE controller. 0 = DISABLE 1 = ENABLE
5	ENABLE	SWR_SDMMC3_RST: Reset SDMMC3 controller. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	ENABLE	SWR_SBC4_RST: Reset SBC4 (SPI 4) controller. 0 = DISABLE 1 = ENABLE
3	ENABLE	SWR_I2C3_RST: Reset I2C3 controller. 0 = DISABLE 1 = ENABLE
2	ENABLE	SWR_UART5_RST: Reset UART5 controller. 0 = DISABLE 1 = ENABLE
1	ENABLE	SWR_UART4_RST: Reset UART4 controller. 0 = DISABLE 1 = ENABLE
0	ENABLE	SWR_SPEEDO_RST: Reset SPEEDO controller. 0 = DISABLE 1 = ENABLE

5.4.5 CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0

Offset: 010h | Read/Write: R/W | Reset: 0b1x00000000000000000000100110xx0

Bit	Reset	Description
31	ENABLE	CLK_ENB_CACHE2: Enable clock to COP cache controller. 0 = DISABLE 1 = ENABLE
29	DISABLE	CLK_ENB_VCP: Enable clock to vector co-processor. 0 = DISABLE 1 = ENABLE
28	DISABLE	CLK_ENB_HOST1X: Enable clock to HOST1X. 0 = DISABLE 1 = ENABLE
27	DISABLE	CLK_ENB_DISP1: Enable clock to DISP1 controller. 0 = DISABLE 1 = ENABLE
26	DISABLE	CLK_ENB_DISP2: Enable clock to DISP2 controller. 0 = DISABLE 1 = ENABLE
25	DISABLE	CLK_ENB_IDE: Enable clock to IDE controller 0 = DISABLE 1 = ENABLE
24	DISABLE	CLK_ENB_3D: Enable clock to 3D controller. 0 = DISABLE 1 = ENABLE
23	DISABLE	CLK_ENB_ISP: Enable clock to ISP controller. 0 = DISABLE 1 = ENABLE
22	DISABLE	CLK_ENB_USBD: Enable clock to USB controller 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
21	DISABLE	CLK_ENB_2D: Enable clock to 2D graphics engine. 0 = DISABLE 1 = ENABLE
20	DISABLE	CLK_ENB_VI: Enable clock to VI controller. 0 = DISABLE 1 = ENABLE
19	DISABLE	CLK_ENB_EPP: Enable clock to EPP controller. 0 = DISABLE 1 = ENABLE
18	DISABLE	CLK_ENB_I2S2: Enable clock to I2S 2 controller 0 = DISABLE 1 = ENABLE
17	DISABLE	CLK_ENB_PWM: Enable clock to PWM (Pulse Width Modulator) 0 = DISABLE 1 = ENABLE
16	DISABLE	CLK_ENB_TWC: Enable clock to 3-Wire Interface Controller 0 = DISABLE 1 = ENABLE
15	DISABLE	CLK_ENB_SDMMC4: Enable clock to SDMMC4 controller. 0 = DISABLE 1 = ENABLE
14	DISABLE	CLK_ENB_SDMMC1: Enable clock to SDMMC1 controller. 0 = DISABLE 1 = ENABLE
13	DISABLE	CLK_ENB_NDFLASH: Enable clock to NAND flash controller. 0 = DISABLE 1 = ENABLE
12	DISABLE	CLK_ENB_I2C1: Enable clock to I2C1 Controller 0 = DISABLE 1 = ENABLE
11	DISABLE	CLK_ENB_I2S1: Enable clock to I2S1 Controller 0 = DISABLE 1 = ENABLE
10	DISABLE	CLK_ENB_SPDIF: Enable clock to SPDIF Controller 0 = DISABLE 1 = ENABLE
9	DISABLE	CLK_ENB_SDMMC2: Enable clock to SDMMC2 controller. 0 = DISABLE 1 = ENABLE
8	ENABLE	CLK_ENB_GPIO: Enable clock to GPIO Controller 0 = DISABLE 1 = ENABLE
7	DISABLE	CLK_ENB_UART2: Enable clock to UART2/VFIR Controller 0 = DISABLE 1 = ENABLE
6	DISABLE	CLK_ENB_UART1: Enable clock to UART1 Controller 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
5	ENABLE	CLK_ENB_TMR: Enable clock to Timer Controller 0 = DISABLE 1 = ENABLE
4	ENABLE	CLK_ENB_RTC: Enable clock to RTC Controller 0 = DISABLE 1 = ENABLE
3	DISABLE	CLK_ENB_AC97: Enable clock to AC97 Controller 0 = DISABLE 1 = ENABLE
0	DISABLE	CLK_ENB_CPU: Enable clock to CPU. 0 = DISABLE 1 = ENABLE

5.4.6 CLK_RST_CONTROLLER_CLK_OUT_ENB_H_0

Offset: 014h | Read/Write: R/W | Reset: 0b0000000x00000000000001001000x000

Bit	Reset	Description
31	DISABLE	CLK_ENB_BSEV: Enable clock to BSEV Controller. 0 = DISABLE 1 = ENABLE
30	DISABLE	CLK_ENB_BSEA: Enable clock to BSEA Controller 0 = DISABLE 1 = ENABLE
29	DISABLE	CLK_ENB_VDE: Enable clock to VDE Controller 0 = DISABLE 1 = ENABLE
28	DISABLE	CLK_ENB_MPE: Enable clock to MPE controller. 0 = DISABLE 1 = ENABLE
27	DISABLE	CLK_ENB_USB3: Enable clock to USB3 controller. 0 = DISABLE 1 = ENABLE
26	DISABLE	CLK_ENB_USB2: Enable clock to USB2 controller. 0 = DISABLE 1 = ENABLE
25	DISABLE	CLK_ENB_EMC: Enable clock to EMC controller. 0 = DISABLE 1 = ENABLE
23	DISABLE	CLK_ENB_UART3: Enable clock to UART3 Controller 0 = DISABLE 1 = ENABLE
22	DISABLE	CLK_ENB_I2C2: Enable clock to I2C2 controller. 0 = DISABLE 1 = ENABLE
21	DISABLE	CLK_ENB_TVDAC: Enable clock to TVDAC controller. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20	DISABLE	CLK_ENB_CSI: Enable clock to CSI controller 0 = DISABLE 1 = ENABLE
19	DISABLE	CLK_ENB_HDMI: Enable clock to HDMI 0 = DISABLE 1 = ENABLE
18	DISABLE	CLK_ENB_MIPI: Enable clock to MIPI base-band controller 0 = DISABLE 1 = ENABLE
17	DISABLE	CLK_ENB_TVO: Enable clock to TVO/CVE controller 0 = DISABLE 1 = ENABLE
16	DISABLE	CLK_ENB_DSI: Enable clock to DSI controller 0 = DISABLE 1 = ENABLE
15	DISABLE	CLK_ENB_DVC_I2C: Enable clock to DVC-I2C Controller 0 = DISABLE 1 = ENABLE
14	DISABLE	CLK_ENB_SBC3: Enable clock to SBC 3 (SPI 3) Controller. 0 = DISABLE 1 = ENABLE
13	DISABLE	CLK_ENB_XIO: Enable clock to XIO Controller. 0 = DISABLE 1 = ENABLE
12	DISABLE	CLK_ENB_SBC2: Enable clock to SBC 2 (SPI 2) Controller. 0 = DISABLE 1 = ENABLE
11	DISABLE	CLK_ENB_SPI1: Enable clock to SPI 1 Controller 0 = DISABLE 1 = ENABLE
10	ENABLE	CLK_ENB_SNOR: Enable clock to NOR Flash Controller. 0 = DISABLE 1 = ENABLE
9	DISABLE	CLK_ENB_SBC1: Enable clock to SBC 1 (SPI 1) Controller. 0 = DISABLE 1 = ENABLE
8	DISABLE	CLK_ENB_KFUSE: Enable clock to KFUSE controller. 0 = DISABLE 1 = ENABLE
7	ENABLE	CLK_ENB_FUSE: Enable clock to FUSE controller. 0 = DISABLE 1 = ENABLE
6	DISABLE	CLK_ENB_PMC: Enable clock to PMC controller. 0 = DISABLE 1 = ENABLE
5	DISABLE	CLK_ENB_STAT_MON: Enable clock to statistic monitor. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
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/EMC. 0 = DISABLE 1 = ENABLE

5.4.7 CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0

Offset: 018h | Read/Write: R/W | Reset: 0b000x1111111xxxxxxx01x1000000000

Bit	Reset	Description
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
26	ENABLE	CLK_M_DOUBLER_ENB: Enable CLK_M clk doubler. 0 = DISABLE 1 = ENABLE
25	ENABLE	SYNC_CLK_DOUBLER_ENB: Enable audio sync clk doubler. 0 = DISABLE 1 = ENABLE
24	ENABLE	CLK_ENB_CRAM2: Enable COP cache ram clk. 0 = DISABLE 1 = ENABLE
23	ENABLE	CLK_ENB_IRAMD: Enable IRAMD clk. 0 = DISABLE 1 = ENABLE
22	ENABLE	CLK_ENB_IRAMC: Enable IRAMC clk. 0 = DISABLE 1 = ENABLE
21	ENABLE	CLK_ENB_IRAMB: Enable IRAMB clk. 0 = DISABLE 1 = ENABLE
20	ENABLE	CLK_ENB_IRAMA: Enable IRAMB clk. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
11	ENABLE	CLK_ENB_AVPUCQ: Enable clock to AVPUCQ. 0 = DISABLE 1 = ENABLE
9	ENABLE	CLK_ENB_CSITE: Enable clock to Coresight. 0 = DISABLE 1 = ENABLE
8	DISABLE	CLK_ENB_AFI: Enable clock to AFI. 0 = DISABLE 1 = ENABLE
7	DISABLE	CLK_ENB_OWR: Enable clock to OWR. 0 = DISABLE 1 = ENABLE
6	DISABLE	CLK_ENB_PCIE: Enable clock to PCIE. 0 = DISABLE 1 = ENABLE
5	DISABLE	CLK_ENB_SDMMC3: Enable clock to SDMMC3. 0 = DISABLE 1 = ENABLE
4	DISABLE	CLK_ENB_SBC4: Enable clock to SBC4 (SPI 4). 0 = DISABLE 1 = ENABLE
3	DISABLE	CLK_ENB_I2C3: Enable clock to I2C3. 0 = DISABLE 1 = ENABLE
2	DISABLE	CLK_ENB_UART5: Enable clock to UART5. 0 = DISABLE 1 = ENABLE
1	DISABLE	CLK_ENB_UART4: Enable clock to UART4. 0 = DISABLE 1 = ENABLE
0	DISABLE	CLK_ENB_SPEEDO: Enable clock to SPEEDO. 0 = DISABLE 1 = ENABLE

5.4.8 CLK_RST_CONTROLLER_CCLK_BURST_POLICY_0

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

Here, the Cortex-A9 CPU Complex is referred to as CPU while ARM7 is referred to as COP. The clock control mechanism for CPU and COP/system are identical. Each CCLK and SCLK clock domain can have 5 states. They are SUSP (suspend) where the clock source is 32 kHz, and normal states (IDLE, RUN, IRQ, FIQ). Each of the normal states can be selected by SW from 8 different clock sources. Furthermore, if any of the CPU/COP FIQ/IRQ bit is enabled, a hardware auto trigger feature will be enabled such that HW will jump from any state to IRQ or to FIQ automatically. Of course, if the source is from a PLL, SW needs to guarantee that the PLL clock is running and stable before changing clock sources.

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

- SW can simply just keep changing the clock source to one state (i.e. just CWAKEUP_IDLE_SOURCE) without changing the CPU_STATE field.
- SW can have the concept of multiple states by first setup CWAKEUP__SOURCE and then just keep changing CPU_STATE field.

- SW can enable HW auto detect of IRQ/FIQ to jump to IRQ or FIQ state.

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

CCLK and SCLK Super Clock Divider Control

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

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

- It has no clock source switching penalty. SW can just pick a PLL output as a max frequency source via CCLK/SCLK_BURST_POLICY and just keep changing SUPER_CCLK/SCLK_DIVIDER to yield the desire lower "effective" frequency.
- For application where the "osc" or "osc*2" frequency is more than sufficient to do the job. PLLs can be turned off to save power and super clock divider can further divide down the "osc" or "osc*2" clock to yield even more power saving.
- If auto IRQ/FIQ feature is enabled, CCLK/SCLK will automatically jump back to full frequency to handle high priority interrupt routine.

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

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

Bit	Reset	Description
31:28	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	0x0	COP_AUTO_CWAKEUP_FROM_FIQ: 0 = NOP ; 1=Burst on COP FIQ
26	0x0	CPU_AUTO_CWAKEUP_FROM_FIQ: 0 = NOP ; 1=Burst on CPU FIQ
25	0x0	COP_AUTO_CWAKEUP_FROM_IRQ: 0 = NOP ; 1=Burst on COP IRQ
24	0x0	CPU_AUTO_CWAKEUP_FROM_IRQ: 0 = NOP ; 1=Burst on CPU IRQ

Bit	Reset	Description
15:12	0x0	CWAKEUP_FIQ_SOURCE: 0000 = clk_m, 0001 = pllC_out0, 0010 = clk_s, 0011 = pllM_out0, 0100 = pllP_out0, 0101 = pllP_out4, 0110 = pllP_out3, 0111 = clk_d, 1xxx = PLLX_out0, 0 = CLKM 1 = PLLC_OUT0 2 = CLKS 3 = PLLM_OUT0 4 = PLLP_OUT0 5 = PLLP_OUT4 6 = PLLP_OUT3 7 = CLKD 8 = PLLX_OUT0
11:8	0x0	CWAKEUP_IRQ_SOURCE: Same definitions as CWAKEUP_FIQ_SOURCE 0 = CLKM 1 = PLLC_OUT0 2 = CLKS 3 = PLLM_OUT0 4 = PLLP_OUT0 5 = PLLP_OUT4 6 = PLLP_OUT3 7 = CLKD 8 = PLLX_OUT0
7:4	0x0	CWAKEUP_RUN_SOURCE: Same definitions as CWAKEUP_FIQ_SOURCE 0 = CLKM 1 = PLLC_OUT0 2 = CLKS 3 = PLLM_OUT0 4 = PLLP_OUT0 5 = PLLP_OUT4 6 = PLLP_OUT3 7 = CLKD 8 = PLLX_OUT0
3:0	0x0	CWAKEUP_IDLE_SOURCE: Same definitions as CWAKEUP_FIQ 0 = CLKM 1 = PLLC_OUT0 2 = CLKS 3 = PLLM_OUT0 4 = PLLP_OUT0 5 = PLLP_OUT4 6 = PLLP_OUT3 7 = CLKD 8 = PLLX_OUT0

5.4.9 CLK_RST_CONTROLLER_SUPER_CCLK_DIVIDER_0

Offset: 024h | Read/Write: R/W | Reset: 0b0xxx0000xxxxxxxx0000000000000000

Bit	Reset	Description
31	DISABLE	SUPER_CDIV_ENB: 0 = disable divider. 0 = DISABLE 1 = ENABLE
27	0x0	SUPER_CDIV_DIS_FROM_COP_FIQ: 0 = enable FIQ, disable FIQ.
26	0x0	SUPER_CDIV_DIS_FROM_CPU_FIQ: 0 = enable FIQ, disable FIQ.
25	0x0	SUPER_CDIV_DIS_FROM_COP_IRQ: 0 = enable IRQ, disable IRQ.
24	0x0	SUPER_CDIV_DIS_FROM_CPU_IRQ: 0 = enable IRQ, disable IRQ.
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = n + 1.

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

5.4.10 CLK_RST_CONTROLLER_SCLK_BURST_POLICY_0

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

Bit	Reset	Description
31:28	0x1	SYS_STATE: 0000=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
14:12	0x0	SWAKEUP_FIQ_SOURCE: 000 = clk_m, 001 = pllC_out1, 010 = pllP_out4, 011 = pllP_out3, 100 = pllP_out2, 101 = clk_d, 110 = clk_s, 111 = pllM_out1, 0 = CLKM 1 = PLLC_OUT1 2 = PLLP_OUT4 3 = PLLP_OUT3 4 = PLLP_OUT2 5 = CLKD 6 = CLKS 7 = PLLM_OUT1
10:8	0x0	SWAKEUP_IRQ_SOURCE: Same definitions as SWAKEUP_FIQ_SOURCE 0 = CLKM 1 = PLLC_OUT1 2 = PLLP_OUT4 3 = PLLP_OUT3 4 = PLLP_OUT2 5 = CLKD 6 = CLKS 7 = PLLM_OUT1
6:4	0x0	SWAKEUP_RUN_SOURCE: Same definitions as SWAKEUP_FIQ_SOURCE 0 = CLKM 1 = PLLC_OUT1 2 = PLLP_OUT4 3 = PLLP_OUT3 4 = PLLP_OUT2 5 = CLKD 6 = CLKS 7 = PLLM_OUT1
2:0	0x0	SWAKEUP_IDLE_SOURCE: Same definitions as SWAKEUP_FIQ_SOURCE 0 = CLKM 1 = PLLC_OUT1 2 = PLLP_OUT4 3 = PLLP_OUT3 4 = PLLP_OUT2 5 = CLKD 6 = CLKS 7 = PLLM_OUT1

5.4.11 CLK_RST_CONTROLLER_SUPER_SCLK_DIVIDER_0

Offset: 02ch | Read/Write: R/W | Reset: 0b0xxx0000xxxxxxxx0000000000000000

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

5.4.12 CLK_RST_CONTROLLER_CLK_SYSTEM_RATE_0

Audio Sync Clock

The purpose of the audio sync clock is to synchronize communication between 2 audio devices in the Tegra 2 Series devices so that it won't get into a FIFO overrun/underrun problem in either of the audio devices. For example, one can receive data from SPDIFIN and transmit the same data back out to an I2S speaker. But if SPDIFIN stream is 43.9 kHz while we are transmitting 44.1 kHz sample rate to I2S, SPDIFIN receive FIFO can get overrun while I2S transmit FIFO can get underrun.

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

HCLK/PCLK

SCLK is the main system clock of the Tegra 2 Series processor which can run up-to 200 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: 030h | Read/Write: R/W | Reset: 0b0x000x00

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

5.4.13 CLK_RST_CONTROLLER_PROG_DLY_CLK_0

Clock Doubler

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

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

Offset: 034h | Read/Write: R/W | Reset: 0b01110111

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

5.4.14 CLK_RST_CONTROLLER_AUDIO_SYNC_CLK_RATE_0

Offset: 038h | Read/Write: R/W | Reset: 0b00000

Bit	Reset	Description
4	0x0	SYNC_CLK_DIS: 0 = Enable AUDIO SYNC CLK
3:0	0x0	SYNC_CLK_RATE: 0000 = SPDIFIN recovered bit clock. 0001 = I2S1 bit clock. 0010 = I2S2 bit clock. 0011 = AC97 bit clock. 0100 = pllA_out0. 0101 = external audio clock (dap_mclk2). 0110 = external audio clock (dap_mclk1). 0111 = external vimclk (vimclk). 1xxx = reserved 0 = SPDIFIN 1 = I2S1 2 = I2S2 3 = AC97 4 = PLLA_OUT0 5 = EXT_AUDIO_CLK2 6 = EXT_AUDIO_CLK1 7 = EXT_VIMCLK

5.4.15 CLK_RST_CONTROLLER_COP_CLK_SKIP_POLICY_0

COP Clock Skip

COP uses the same clock as the system (SCLK). The COP_CLK_SKIP_POLICY register provides a mechanism to slow down the COP without slowing down the system clock. This is done by deferring/delaying the ready signal back to the COP. Again, HW can auto detect CPU/COP IRQ/FIQ and change the skip rate as programmed by SW.

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

Bit	Reset	Description
31:28	0x0	COP_CLK_SKIP_STATE: 0000=no skip. 0001=skip base on IDLE Clock skip rate; 001X=skip base on Run clock skip rate; 01XX=skip base on IRQ Clock skip rate; 1XXX=skip base on FIQ Clock skip rate
27	0x0	COP_CLK_SKIP_ENB_FROM_COP_FIQ: 0 = NOP ; 1=Burst on COP FIQ
26	0x0	COP_CLK_SKIP_ENB_FROM_CPU_FIQ: 0 = NOP ; 1=Burst on CPU FIQ
25	0x0	COP_CLK_SKIP_ENB_FROM_COP_IRQ: 0 = NOP ; 1=Burst on COP IRQ
24	0x0	COP_CLK_SKIP_ENB_FROM_CPU_IRQ: 0 = NOP ; 1=Burst on CPU IRQ

Bit	Reset	Description
14:12	0x0	COP_CLK_SKIP_RATE_FIQ: skip n/16 clock.
10:8	0x0	COP_CLK_SKIP_RATE_IRQ: Same definitions as COP_CLK_SKIP_RATE_FIQ
6:4	0x0	COP_CLK_SKIP_RATE_RUN: Same definitions as COP_CLK_SKIP_RATE_FIQ
2:0	0x0	COP_CLK_SKIP_RATE_IDLE: Same definitions as COP_CLK_SKIP_RATE_FIQ

5.4.16 CLK_RST_CONTROLLER_CLK_MASK_ARM_0

Offset: 044h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxx00xxxxxxxxxxxxxx00

Bit	R/W	Reset	Description
31	RO	X	AXI_FLUSH_DONE: 1 = CPU AXI pipe is flushed.
16	RW	0x0	CLK_MASK_CPU_HALT: 1 = HW will stop clock to CPU when halt, 0 = no clock stop.
1:0	RW	0x0	CLK_MASK_COP: 00 = no clock masking. 01 = u2_nwait_r. 10 = u2_nwait_r. 11 = no clock masking.

5.4.17 CLK_RST_CONTROLLER_MISC_CLK_ENB_0

Offset: 048h | Read/Write: R/W | Reset: 0b0000

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.

5.4.18 CLK_RST_CONTROLLER_CLK_CPU_CMPLX_0

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

Note: Since the CPU clock can be selected from 1-of-9 clock sources (refer to CCLK_BURST_POLICY register), it's the user's responsibility to not select a bridge divide down CPU clock ratio that will exceed 1/4 of the "MAX" CPU frequency supported.
For Tegra 2, the "MAX" CPU frequency is 1.1 GHz.

Offset: 04ch | Read/Write: R/W | Reset: 0b00xxxxxx11

Bit	Reset	Description
9	0x0	CPU1_CLK_STP: 1 = CPU1 clock stop, 0 = CPU1 clock run.
8	0x0	CPU0_CLK_STP: 1 = CPU0 clock stop, 0 = CPU0 clock run.
1:0	0x3	CPU_BRIDGE_CLKDIV: Clock divider ratio for the cpu bridge devices connected to CPU/L2-cache. 00 = div-by-1. 01 = div-by-2. 10 = div-by-3. 11 = div-by-4.

5.4.19 CLK_RST_CONTROLLER_OSC_CTRL_0

Oscillator Control

"osc" can have any of the four frequency (12MHz, 13MHz, 19.2MHz, 26MHz) coming in, the OSC_FREQ field provides a way for SW to tell HW what frequency is the incoming clock runs at. This information is used by hardware to auto setup the parameters (DIVN, DIVM, DIVP, CPCON, LFCON, VCOCON, DCCON, OUT1_RATIO, OUT2_RATIO, OUT3_RATIO, and OUT4_RATIO) to PLLP and its dividers. In case a frequency other than these 4 are used, there is a way to override the HW auto generated PLLP parameters.

Note: Only 12 MHz required for PCIe (Tegra 250 only) operation.

Offset: 050h | Read/Write: R/W | Reset: 0b000000000000xxx00000xx111111xx01

Bit	Reset	Description
31:30	0x0	OSC_FREQ: 00 = 13MHz, 01 = 19.2MHz, 10 = 12MHz, 11 = 26MHz.
29:28	0x0	PLL_REF_DIV: PLL reference clock divide. 00 = /1, 01 = /2, 10 = /4, 11 = reserve.
27:20	0x0	OSCFI_SPARE: Crystal oscillator spare register control.
16:12	0x0	XODS: Crystal oscillator duty cycle control.
9:4	0x3f	XOFS: Crystal oscillator drive strength control.
1	0x0	XOBP: Crystal oscillator bypass enable (1 = enable bypass).
0	0x1	XOE: Crystal oscillator enable (1 = enable).

5.4.20 CLK_RST_CONTROLLER_PLL_LFSR_0

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

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

5.4.21 CLK_RST_CONTROLLER_OSC_FREQ_DET_0

Oscillator Frequency Detect

"osc" can have any one of the four frequency (12 MHz, 13 MHz, 19.2 MHz, 26 MHz) coming in, HW is providing a mechanism of detecting which one of the frequencies the "osc" is running at. To determine the "osc" frequency, user only needs to program the number of 32 kHz (actually, 32.768 kHz) clock periods use as a fix time window from which the "osc" will be used to increment a 16-bit counter and start the frequency detection process. Once the detection process is done, user can just check the 16-bit count value to determine the "osc" frequency.

OSC Frequency (MHz)	Approximate OSC_FREQ_DET_CNT (using two 32 kHz period as window)
12.00	732
13.00	794
19.20	1172
26.00	1587

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

Bit	Reset	Description
31	DISABLE	OSC_FREQ_DET_TRIG: 0 = default, 1 = enable osc frequency detect. 0 = DISABLE 1 = ENABLE
3:0	0x0	REF_CLK_WIN_CFG: Indicate the # of 32 kHz clock period as window in n+1 scheme.

5.4.22 CLK_RST_CONTROLLER_OSC_FREQ_DET_STATUS_0

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

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

5.4.23 CLK_RST_CONTROLLER_PLLC_BASE_0

PLL Configuration

Tegra 2 has 9 PLLs controllable by clock control.

- "PLLC/PLLP/PLLA" is of type CLKPLL700_LP.
- "PLLD" is of type CLKPLL1G_MIPI.
- "PLLU" is of type CLKPLL960_USB.
- "PLLM/PLLX" are of type CLKPLL12G_LP.
- "PLLE" is of type PLL24G_SSA_CML_ESD_LP (PLLE is supported only on Tegra 250).
- "PLLS" is of type CLKPLL32K_LP

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

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

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

Table 15 PLL Requirements for software

	CLKPLL700_LP	CLKPLL1G_MIPI	CLKPLL12G_LP	CLKPLL960_USB
Input freq. (REF)	2-31 MHz	2-40 MHz	2-31 MHz	2-40 MHz
Comparison freq. (CF)	1-6 MHz	1-6 MHz	1-6 MHz	1-6 MHz
VCO freq. (VCO)	20-1400 MHz	40-1000 MHz	20-1200 MHz	480-960 MHz

Table 16 Charge pump current (CPCON) for CLKPLL700_LP/CLKPLL12G_LP

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

Table 17 Charge pump current (CPCON) for CLKPLL1G_MIPI/CLKPLL960_USB

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

Table 18 PLLP Configuration Information (reference clock is osc/clk_m and PLLP-FO is fixed at 432 MHz).

Reference Frequency	13.0 MHz	19.2 MHz	12.0 MHz	26.0 MHz	Resulting Frequency
DIVN	432 (1b0h)	90 (05ah)	432 (1b0h)	432 (1b0h)	N/A
DIVM	13 (0dh)	4 (04h)	12 (0ch)	26 (1ah)	N/A
DIVP	1 (0h)	1 (0h)	1 (0h)	1 (0h)	N/A
CPCON	8 (8h)	1 (1h)	8 (8h)	8 (8h)	N/A
LFCON	0 (0h)	0 (0h)	0 (0h)	0 (0h)	N/A
VCOCON	0 (0h)	0 (0h)	0 (0h)	0 (0h)	N/A
DCCON	0 (0b)	0 (0b)	0 (0b)	0 (0b)	N/A
OUT1 div ratio	15.0 (1ch)	15.0(1ch)	15.0(1ch)	15.0(1ch)	28.8MHz.
OUT2 div ratio	9.0 (10h)	9.0(10h)	9.0(10h)	9.0(10h)	48.0MHz.
OUT3 div ratio	6.0 (0ah)	6.0(0ah)	6.0(0ah)	6.0(0ah)	72.0MHz.
OUT4 div ratio	4.0 (06h)	4.0(06h)	4.0(06h)	4.0(06h)	108.0MHz.

Table 19 PLLA configuration information (reference clock is taken from PLLP_OUT1 = 28.8 MHz).

Audio Frequency	56.448 MHz	73.728 MHz	11.2896 MHz	12.2880 MHz
DIVN	49 (031h)	64 (040h)	49 (031h)	64 (040h)

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

Table 20 PLLU configuration information (reference clock is osc/clk_m and PLLU-FOs are fixed at 12 MHz/60 MHz/480 MHz).

Reference Frequency	13.0 MHz	19.2 MHz	12.0 MHz	26.0 MHz
DIVN	960 (3c0h)	200 (0c8h)	960 (3c0h)	960 (3c0h)
DIVM	13 (0dh)	4 (04h)	12 (0ch)	26 (1ah)

Table 21 PLLD configuration information (reference clock is osc/clk_m and PLLD-FO is max of 1GHz).

Reference Frequency	13.0 MHz	19.2 MHz	12.0 MHz	26.0 MHz
DIVN	1000(3e8h)	625 (271h)	1000(3e8h)	1000(3e8h)
DIVM	13 (0dh)	12 (0ch)	12 (0ch)	26 (1ah)
DIVP	1 (0h)	1 (0h)	1 (0h)	1 (0h)

Table 22 PLLE configuration information (reference clock is osc/clk_m and PLLE-FO is 100 MHz).

Reference Frequency	12.0 MHz
NDIV	200 (c8h)
MDIV	1 (0h)
PLDIV	24 (18h)
PLDIV_CML	(dh)

Table 23 PLLS configuration information (reference clock is 32.768 kHz).

Reference Frequency	13.0 MHz	19.2 MHz	12.0 MHz	26.0 MHz
DIVN	397 (18dh)	586 (24ah)	366 (16eh)	793 (319h)
DIVM	1 (1h)	1 (1h)	1 (1h)	1 (1h)
DIVP	1 (0h)	1 (0h)	1 (0h)	1 (0h)

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

(1) PLLX is physically located at the same power partition as the CPU and thus will be powered down when the CPU partition is powered-down. Therefore, if PLLX is used as a clock source for the CPU before power-down, then PLLX must be re-initialized after the CPU is powered back up.

(2) If hardware detects that PLLX is used as a clock source for the CPU when CPU partition is powering down, it will reset the following registers to its default value automatically.

- CCLK_BURST_POLICY.

- SUPER_CCLK_DIVIDER.
- PLLX_BASE.
- PLLX_MISC.

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

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLC_BYPASS: 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLC_ENABLE: 0 = disable, 1 = enable. 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLC_REF_DIS: 0 = enable reference clk, 1 = disable reference clk. 0 = REF_ENABLE 1 = REF_DISABLE
27	RO	X	PLLC_LOCK: 0 = not lock, 1 = lock.
22:20	RW	0x0	PLLC_DIVP: 0 = post divider (2^n).
17:8	RW	0x1	PLLC_DIVN: PLL feedback divider.
4:0	RW	0xc	PLLC_DIVM: PLL input divider.

5.4.24 CLK_RST_CONTROLLER_PLLC_OUT_0

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

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

5.4.25 CLK_RST_CONTROLLER_PLLC_MISC_0

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

Bit	Reset	Description
31	0x0	PLLC_OUT1_INV_CLK: 1 = invert PLLC_OUT1 clock.
30	0x0	PLLC_OUT1_DIV_BYP: 1 = bypass PLLC_OUT1 divider.
23:22	0x0	PLLC_PTS: Base PLLC test output select.
20	0x0	PLLC_DCCON: Base PLLC DCCON control.
18	0x0	PLLC_LOCK_ENABLE: 1 = enable, 0 = disable. 0 = DISABLE 1 = ENABLE
17:12	0x0	PLLC_LOCK_SEL: lock select.

Bit	Reset	Description
11:8	0x1	PLLC_CPCON: Base PLLC charge pump setup control.
7:4	0x0	PLLC_LFCON: Base PLLC loop filter setup control.
3:0	0x0	PLLC_VCOCON: Base PLLC VCO range setup control.

5.4.26 CLK_RST_CONTROLLER_PLLM_BASE_0

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

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

5.4.27 CLK_RST_CONTROLLER_PLLM_OUT_0

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

Bit	Reset	Description
15:8	0x0	PLLM_OUT1_RATIO: PLLM_OUT1 divider from base PLLM (lsb denote 0.5x).
1	ENABLE	PLLM_OUT1_CLKEN: PLLM_OUT1 divider clk enable. 0 = disable, 1 = enable. 0 = DISABLE 1 = ENABLE
0	RESET_DISABLE	PLLM_OUT1_RSTN: PLLM_OUT1 divider reset. 0 = reset, 1 = not reset. 0 = RESET_ENABLE 1 = RESET_DISABLE

5.4.28 CLK_RST_CONTROLLER_PLLM_MISC_0

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

Bit	Reset	Description
31	0x0	PLLM_OUT1_INV_CLK: 1 = invert PLLM_OUT1 clock.
30	0x0	PLLM_OUT1_DIV_BYP: 1 = bypass PLLM_OUT1 divider.
27:24	0x0	PLLM_SETUP: Base PLLM setup.

Bit	Reset	Description
23:22	0x0	PLLM_PTS: Base PLLM test output select.
20	0x0	PLLM_DCCON: Base PLLM DCCON control.
18	0x0	PLLM_LOCK_ENABLE: 1 = enable, 0 = disable. 0 = DISABLE 1 = ENABLE
17:12	0x0	PLLM_LOCK_SEL: lock select.
11:8	0x1	PLLM_CPCON: Base PLLM charge pump setup control.
7:4	0x0	PLLM_LFCON: Base PLLM loop filter setup control.
3:0	0x0	PLLM_VCOCON: Base PLLM VCO range setup control.

5.4.29 CLK_RST_CONTROLLER_PLLP_BASE_0

Offset: 0a0h | Read/Write: R/W | Reset: 0b0000xxxxx000xx0000000001xxx01100

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLP_BYPASS: 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLP_ENABLE: 0 = disable, 1 = enable. 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLP_REF_DIS: 0 = enable reference clk, 1 = disable reference clk. 0 = REF_ENABLE 1 = REF_DISABLE
28	RW	DISABLE	PLLP_BASE_OVRRIDE: 0 = disallow base override , 1 = allow base override. 0 = DISABLE 1 = ENABLE
27	RO	X	PLLP_LOCK: 0 = not lock, 1 = lock.
22:20	RW	0x0	PLLP_DIVP: 0 = post divider (2^n).
17:8	RW	0x1	PLLP_DIVN: PLL feedback divider.
4:0	RW	0xc	PLLP_DIVM: PLL input divider.

5.4.30 CLK_RST_CONTROLLER_PLLP_OUTA_0

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

Bit	Reset	Description
31:24	0x0	PLLP_OUT2_RATIO: PLLP_OUT2 divider from base PLLP (lsb denote 0.5x).
18	DISABLE	PLLP_OUT2_OVRRIDE: 0 = disallow PLLP_OUT2 ratio override, 1 = enable override. 0 = DISABLE 1 = ENABLE
17	ENABLE	PLLP_OUT2_CLKEN: PLLP_OUT2 divider clk enable. 0 = disable, 1 = enable. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
16	RESET_DISABLE	PLLP_OUT2_RSTN: PLLP_OUT2 divider reset. 0 = reset, 1 = not reset. 0 = RESET_ENABLE 1 = RESET_DISABLE
15:8	0x0	PLLP_OUT1_RATIO: PLLP_OUT1 divider from base PLLP (lsb denote 0.5x).
2	DISABLE	PLLP_OUT1_OVRRIDE: 0 = disallow PLLP_OUT1 ratio override, 1 = enable override. 0 = DISABLE 1 = ENABLE
1	ENABLE	PLLP_OUT1_CLKEN: PLLP_OUT1 divider clk enable. 0 = disable, 1 = enable. 0 = DISABLE 1 = ENABLE
0	RESET_DISABLE	PLLP_OUT1_RSTN: PLLP_OUT1 divider reset. 0 = reset, 1 = not reset. 0 = RESET_ENABLE 1 = RESET_DISABLE

5.4.31 CLK_RST_CONTROLLER_PLLP_OUTB_0

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

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

5.4.32 CLK_RST_CONTROLLER_PLLP_MISC_0

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

Bit	Reset	Description
31	0x0	PLLP_OUT4_INV_CLK: 1 = invert PLLP_OUT4 clock.

Bit	Reset	Description
30	0x0	PLLP_OUT3_INV_CLK: 1 = invert PLLP_OUT3 clock.
29	0x0	PLLP_OUT2_INV_CLK: 1 = invert PLLP_OUT2 clock.
28	0x0	PLLP_OUT1_INV_CLK: 1 = invert PLLP_OUT1 clock.
27	0x0	PLLP_OUT4_DIV_BYP: 1 = bypass PLLP_OUT4 divider.
26	0x0	PLLP_OUT3_DIV_BYP: 1 = bypass PLLP_OUT3 divider.
25	0x0	PLLP_OUT2_DIV_BYP: 1 = bypass PLLP_OUT2 divider.
24	0x0	PLLP_OUT1_DIV_BYP: 1 = bypass PLLP_OUT1 divider.
23:22	0x0	PLLP_PTS: Base PLLP test output select.
20	0x0	PLLP_DCCON: Base PLLP DCCON control.
18	0x0	PLLP_LOCK_ENABLE: 1 = enable, 0 = disable. 0 = DISABLE 1 = ENABLE
17:12	0x0	PLLP_LOCK_SEL: lock select.
11:8	0x1	PLLP_CPCON: Base PLLP charge pump setup control.
7:4	0x0	PLLP_LFCON: Base PLLP loop filter setup control.
3:0	0x0	PLLP_VCOCON: Base PLLP VCO range setup control.

5.4.33 CLK_RST_CONTROLLER_PLLA_BASE_0

Offset: 0b0h | Read/Write: R/W | Reset: 0b000xxxxxx000xx0000000001xxx01100

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

5.4.34 CLK_RST_CONTROLLER_PLLA_OUT_0

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

Bit	Reset	Description
15:8	0x0	PLLA_OUT0_RATIO: PLLA_OUT0 divider from base PLLA (lsb denote 0.5x). 0 = DISABLE 1 = ENABLE
1	ENABLE	PLLA_OUT0_CLKEN: PLLA_OUT0 divider clk enable. 0 = disable, 1 = enable. 0 = DISABLE 1 = ENABLE
0	RESET_DISABLE	PLLA_OUT0_RSTN: PLLA_OUT0 divider reset. 0 = reset, 1 = not reset. 0 = RESET_ENABLE 1 = RESET_DISABLE

5.4.35 CLK_RST_CONTROLLER_PLLA_MISC_0

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

Bit	Reset	Description
31	0x0	PLLA_OUT0_INV_CLK: 1 = invert PLLA_OUT0 clock.
30	0x0	PLLA_OUT0_DIV_BYP: 1 = bypass PLLA_OUT0 divider.
23:22	0x0	PLLA_PTS: Base PLLA test output select.
20	0x0	PLLA_DCCON: Base PLLA DCCON control.
18	0x0	PLLA_LOCK_ENABLE: 1 = enable, 0 = disable. 0 = DISABLE 1 = ENABLE
17:12	0x0	PLLA_LOCK_SEL: lock select.
11:8	0x1	PLLA_CPCON: Base PLLA charge pump setup control.
7:4	0x0	PLLA_LFCON: Base PLLA loop filter setup control.
3:0	0x0	PLLA_VCOCON: Base PLLA VCO range setup control.

5.4.36 CLK_RST_CONTROLLER_PLLU_BASE_0

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

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLU_BYPASS: 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLU_ENABLE: 0 = disable, 1 = enable. This bit is used only when PLLU_OVERRIDE bit is set. 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLU_REF_DIS: 0 = enable reference clk, 1 = disable reference clk. 0 = REF_ENABLE 1 = REF_DISABLE

Bit	R/W	Reset	Description
27	RO	X	PLLU_LOCK: 0 = not lock, 1 = lock.
23	RW	0x0	PLLU_CLKENABLE_ICUSB: FO_ICUSB output enable. This bit is use only when PLLU_OVERRIDE bit is set.
22	RW	0x0	PLLU_CLKENABLE_HSIC: FO_HSIC output enable. This bit is used only when PLLU_OVERRIDE bit is set. Otherwise, USB controllers will control this automatically. 0 = disable, 1 = enable.
21	RW	0x0	PLLU_CLKENABLE_USB: FO_USB output enable. This bit is used only when PLLU_OVERRIDE bit is set. Otherwise, USB controllers will control this automatically. 0 = disable, 1 = enable.
20	RW	0x0	PLLU_VCO_FREQ: 0 = post-div of 2, 1 = post-div of 1.
17:8	RW	0x1	PLLU_DIVN: PLL feedback divider.
4:0	RW	0xc	PLLU_DIVM: PLL input divider.

5.4.37 CLK_RST_CONTROLLER_PLLU_MISC_0

Offset: 0cch | Read/Write: R/W | Reset: 0b000xxxx0xxxx000000000100000000

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

5.4.38 CLK_RST_CONTROLLER_PLLD_BASE_0

Offset: 0d0h | Read/Write: R/W | Reset: 0b000xxxxxx000xx0000000001xxx01100

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLD_BYPASS: 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLD_ENABLE: 0 = disable, 1 = enable. 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLD_REF_DIS: 0 = enable reference clk, 1 = disable reference clk. 0 = REF_ENABLE 1 = REF_DISABLE
27	RO	X	PLLD_LOCK: 0 = not lock, 1 = lock.
22:20	RW	0x0	PLLD_DIVP: 0 = post divider (2^n).
17:8	RW	0x1	PLLD_DIVN: PLL feedback divider.

Bit	R/W	Reset	Description
4:0	RW	0xc	PLLD_DIVM: PLL input divider.

5.4.39 CLK_RST_CONTROLLER_PLLD_MISC_0

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

Bit	Reset	Description
31	0x0	PLLD_FO_MODE: 1 = 5-125MHz, 0 = 40-1000MHz.
30	0x0	PLLD_CLKENABLE: 0 = disable, 1 = normal operation.
29:27	0x0	PLLD_PTS: Base PLLD test output select.
26:24	0x0	PLLD_LOADADJ: load pulse position adjust.
23	0x0	PLLD_DIV_RST: 0 = normal operation, 1 = reset.
22	0x0	PLLD_LOCK_ENABLE: 1 = enable, 0 = disable. 0 = DISABLE 1 = ENABLE
21:16	0x0	PLLD_LOCK_SEL: lock select.
15:12	0x0	PLLD_DCCON: Base PLLD DCCON control.
11:8	0x1	PLLD_CPCON: Base PLLD charge pump setup control.
7:4	0x0	PLLD_LFCON: Base PLLD loop filter setup control.
3:0	0x0	PLLD_VCOCON: Base PLLD VCO range setup control.

5.4.40 CLK_RST_CONTROLLER_PLLX_BASE_0

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

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

5.4.41 CLK_RST_CONTROLLER_PLLX_MISC_0

Offset: 0e4h | Read/Write: R/W | Reset: 0b000000x1x0000000000100000000

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

5.4.42 CLK_RST_CONTROLLER_PLLE_BASE_0 (Tegra 250 Only)

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

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

5.4.43 CLK_RST_CONTROLLER_PLLE_MISC_0 (Tegra 250 Only)

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

Bit	R/W	Reset	Description
31:16	RW	0x0	PLLE_SETUP: Base PLLE setup[15:0].
15	RO	X	PLLE_PLL_READY: When read, this is PLL_READY status: 1 = PLL finish training, 0 = PLL not finish training.

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

5.4.44 CLK_RST_CONTROLLER_PLLS_BASE_0

Offset: 0f0h | Read/Write: R/W | Reset: 0b000xxxxxx000xx0000000001xxxx0001

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLS_BYPASS: 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLS_ENABLE: 0 = disable, 1 = enable. 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLS_REF_DIS: 0 = enable reference clk, 1 = disable reference clk. 0 = REF_ENABLE 1 = REF_DISABLE
27	RO	X	PLLS_LOCK: 0 = not lock, 1 = lock.
22:20	RW	0x0	PLLS_DIVP: 0 = post divider (2^n).
17:8	RW	0x1	PLLS_DIVN: PLL feedback divider.
3:0	RW	0x1	PLLS_DIVM: PLL input divider.

5.4.45 CLK_RST_CONTROLLER_PLLS_MISC_0

Offset: 0f4h | Read/Write: R/W | Reset: 0b00xxx00000000000100000000

Bit	Reset	Description
23:22	0x0	PLLS_PTS: Base PLLS test output select.

Bit	Reset	Description
18	0x0	PLLS_LOCK_ENABLE: 1 = enable, 0 = disable. 0 = DISABLE 1 = ENABLE
17:12	0x0	PLLS_LOCK_SEL: lock select.
11:8	0x1	PLLS_CPCON: Base PLLS charge pump setup control.
7:4	0x0	PLLS_LFCON: Base PLLS loop filter setup control.
3:0	0x0	PLLS_VCOCON: Base PLLS VCO range setup control.

5.4.46 CLK_RST_CONTROLLER_CLK_SOURCE_I2S1_0

Clock Configuration for Each Peripherals

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

Note: The 16-bit UART1/UART2/UART3 clock divider control is provided by the UART register set themselves and not by the CLK_SOURCE_ registers.
In order to switch from one clock source to the next, both clock sources must be active/running.

Special PLL clock usage by certain peripheral

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

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

Module/Logic	Fix PLL Clock Source Used
LFSR	pllM_out0
DSI	pllP_out3
CSI	pllP_out3
I2C1	pllP_out3
I2C2	pllP_out3
I2C3	pllP_out3
UART1	pllP_out3
UART2	pllP_out3
UART3	pllP_out3
UART4	pllP_out3
UART5	pllP_out3

Offset: 100h | Read/Write: R/W | | Reset: 0b11x1xxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	I2S1_CLK_SRC: 00 = pllA_out0 01 = audio SYNC_CLK x 2 10 = pllP_out0 11 = clk_m 0 = PLLA_OUT0 1 = SYNC_CLK_X2 2 = PLLP_OUT0 3 = CLK_M
28	0x1	I2S1_MASTER_CLKEN: 1 = enable I2S1 master clock, disable I2S1 master clock.
7:0	0x0	I2S1_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

5.4.47 CLK_RST_CONTROLLER_CLK_SOURCE_I2S2_0

Offset: 104h | Read/Write: R/W | Reset: 0b11x1xxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	I2S2_CLK_SRC: 00 = pllA_out0 01 = audio SYNC_CLK x 2 10 = pllP_out0 11 = clk_m 0 = PLLA_OUT0 1 = SYNC_CLK_X2 2 = PLLP_OUT0 3 = CLK_M
28	0x1	I2S2_MASTER_CLKEN: 1 = enable I2S2 master clock, disable I2S2 master clock.
7:0	0x0	I2S2_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

5.4.48 CLK_RST_CONTROLLER_CLK_SOURCE_SPDIF_OUT_0

Offset: 108h | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	SPDIFOUT_CLK_SRC: 00 = pllA_out0 01 = audio SYNC_CLK x 2 10 = pllP_out0 11 = clk_m 0 = PLLA_OUT0 1 = SYNC_CLK_X2 2 = PLLP_OUT0 3 = CLK_M
7:0	0x0	SPDIFOUT_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

5.4.49 CLK_RST_CONTROLLER_CLK_SOURCE_SPDIF_IN_0

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

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

5.4.50 CLK_RST_CONTROLLER_CLK_SOURCE_PWM_0

Offset: 110h | Read/Write: R/W | Reset: 0b011xxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
30:28	CLK_M	PWM_CLK_SRC: 000 = pllP_out0 001 = pllC_out0 010 = audio SYNC_CLK x 2 011 = clk_m 100 = clk_s 0 = PLLP_OUT0 1 = PLLC_OUT0 2 = SYNC_CLK_X2 3 = CLK_M 4 = CLK_S
7:0	0x0	PWM_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

5.4.51 CLK_RST_CONTROLLER_CLK_SOURCE_SPI1_0

Offset: 114h | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	SPI1_CLK_SRC: 00 = pllP_out0 01 = pllC_out0 10 = pllM_out0 11 = clk_m 0 = PLLP_OUT0 1 = PLLC_OUT0 2 = PLLM_OUT0 3 = CLK_M
7:0	0x0	SPI1_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

5.4.52 CLK_RST_CONTROLLER_CLK_SOURCE_SPI22_0

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

Bit	Reset	Description
31:30	CLK_M	SPI2_CLK_SRC: 00 = pllP_out0 01 = pllC_out0 10 = pllM_out0 11 = clk_m 0 = PLLP_OUT0 1 = PLLC_OUT0 2 = PLLM_OUT0 3 = CLK_M
7:0	0x0	SPI2_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

5.4.53 CLK_RST_CONTROLLER_CLK_SOURCE_SPI3_0

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

Bit	Reset	Description
31:30	CLK_M	SPI3_CLK_SRC: 00 = pllP_out0 01 = pllC_out0 10 = pllM_out0 11 = clk_m 0 = PLLP_OUT0 1 = PLLC_OUT0 2 = PLLM_OUT0 3 = CLK_M
7:0	0x0	SPI3_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

5.4.54 CLK_RST_CONTROLLER_CLK_SOURCE_XIO_0

Offset: 120h | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxxxxxxxxxxx00000000

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

5.4.55 CLK_RST_CONTROLLER_CLK_SOURCE_I2C1_0

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

Bit	Reset	Description
31:30	CLK_M	I2C1_CLK_SRC: 00 = pllP_out0 01 = pllC_out0 10 = pllM_out0 11 = clk_m 0 = PLLP_OUT0 1 = PLLC_OUT0 2 = PLLM_OUT0 3 = CLK_M
15:0	0x0	I2C1_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 1.0x)

5.4.56 CLK_RST_CONTROLLER_CLK_SOURCE_DVC_I2C_0

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

Bit	Reset	Description
31:30	CLK_M	DVC_I2C_CLK_SRC: 00 = pllP_out0 01 = pllC_out0 10 = pllM_out0 11 = clk_m 0 = PLLP_OUT0 1 = PLLC_OUT0 2 = PLLM_OUT0 3 = CLK_M
15:0	0x0	DVC_I2C_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 1.0x)

5.4.57 CLK_RST_CONTROLLER_CLK_SOURCE_TWC_0

Offset: 12ch | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxxxxxxxxxxx00000000

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

5.4.58 CLK_RST_CONTROLLER_CLK_SOURCE_SPI1_0

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

Bit	Reset	Description
31:30	CLK_M	SPI1_CLK_SRC: 00 = pllP_out0 01 = pllC_out0 10 = pllM_out0 11 = clk_m 0 = PLLP_OUT0 1 = PLLC_OUT0 2 = PLLM_OUT0 3 = CLK_M
7:0	0x0	SPI1_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

5.4.59 CLK_RST_CONTROLLER_CLK_SOURCE_DISP1_0

Offset: 138h | Read/Write: R/W | Reset: 0b11

Bit	Reset	Description
31:30	CLK_M	DISP1_CLK_SRC: 00 = pllP_out0 01 = pllD_out0 10 = pllC_out0 11 = clk_m 0 = PLLP_OUT0 1 = PLLD_OUT0 2 = PLLC_OUT0 3 = CLK_M

5.4.60 CLK_RST_CONTROLLER_CLK_SOURCE_DISP2_0

Offset: 13ch | Read/Write: R/W | Reset: 0b11

Bit	Reset	Description
31:30	CLK_M	DISP2_CLK_SRC: 00 = pllP_out0 01 = pllD_out0 10 = pllC_out0 11 = clk_m 0 = PLLP_OUT0 1 = PLLD_OUT0 2 = PLLC_OUT0 3 = CLK_M

5.4.61 CLK_RST_CONTROLLER_CLK_SOURCE_CVE_0

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

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

5.4.62 CLK_RST_CONTROLLER_CLK_SOURCE_IDE_0 (Tegra 250 Only)

Offset: 144h | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxxxxxxxxxxx00000000

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

5.4.63 CLK_RST_CONTROLLER_CLK_SOURCE_VI_0

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

Bit	Reset	Description
31:30	PLLM_OUT0	VI_CLK_SRC: 00 = pllM_out0 01 = pllC_out0 10 = pllP_out0 11 = pllA_out0 0 = PLLM_OUT0 1 = PLLC_OUT0 2 = PLLP_OUT0 3 = PLLA_OUT0
25	0x0	PD2VI_CLK_SEL: 0 = pd2vi_clk, 1 = vi_sensor_clk.
24	INTERNAL	VI_CLK_SEL: 0 = select internal clock, 1 = select external clock (pd2vi_clk). 0 = INTERNAL 1 = EXTERNAL
7:0	0x0	VI_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

5.4.64 CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC1_0

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

Bit	Reset	Description
31:30	CLK_M	SDMMC1_CLK_SRC: 00 = pllP_out0 01 = pllC_out0 10 = pllM_out0 11 = clk_m 0 = PLLP_OUT0 1 = PLLC_OUT0 2 = PLLM_OUT0 3 = CLK_M
7:0	0x0	SDMMC1_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

5.4.65 CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC2_0

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

Bit	Reset	Description
31:30	CLK_M	SDMMC2_CLK_SRC: 00 = pllP_out0 01 = pllC_out0 10 = pllM_out0 11 = clk_m 0 = PLLP_OUT0 1 = PLLC_OUT0 2 = PLLM_OUT0 3 = CLK_M
7:0	0x0	SDMMC2_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

5.4.66 CLK_RST_CONTROLLER_CLK_SOURCE_G3D_0

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

Bit	Reset	Description
31:30	PLLM_OUT0	G3D_CLK_SRC: 00 = pllM_out0 01 = pllC_out0 10 = pllP_out0 11 = pllA_out0 0 = PLLM_OUT0 1 = PLLC_OUT0 2 = PLLP_OUT0 3 = PLLA_OUT0
15:8	0x0	G3D_IDLE_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x) if all 0's, this idle divisor field will not be used. For non-zero values, when host1x is idle, this field will be used instead of G3D_CLK_DIVISOR.
7:0	0x0	G3D_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

5.4.67 CLK_RST_CONTROLLER_CLK_SOURCE_G2D_0

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

Bit	Reset	Description
31:30	PLLM_OUT0	G2D_CLK_SRC: 00 = pllM_out0 01 = pllC_out0 10 = pllP_out0 11 = pllA_out0 0 = PLLM_OUT0 1 = PLLC_OUT0 2 = PLLP_OUT0 3 = PLLA_OUT0
15:8	0x0	G2D_IDLE_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x) if all 0's, this idle divisor field will not be used. For non-zero values, when host1x is idle, this field will be used instead of G2D_CLK_DIVISOR.
7:0	0x0	G2D_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

5.4.68 CLK_RST_CONTROLLER_CLK_SOURCE_NDFLASH_0

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

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

5.4.69 CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC4_0

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

Bit	Reset	Description
31:30	CLK_M	SDMMC4_CLK_SRC: 00 = pllP_out0 01 = pllC_out0 10 = pllM_out0 11 = CLK_M 0 = PLLP_OUT0 1 = PLLC_OUT0 2 = PLLM_OUT0 3 = CLK_M
7:0	0x0	SDMMC4_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

5.4.70 CLK_RST_CONTROLLER_CLK_SOURCE_VFIR_0

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

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

5.4.71 CLK_RST_CONTROLLER_CLK_SOURCE_EPP_0

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

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

5.4.72 CLK_RST_CONTROLLER_CLK_SOURCE_MPE_0

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

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

5.4.73 CLK_RST_CONTROLLER_CLK_SOURCE_MIPI_0

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

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

5.4.74 CLK_RST_CONTROLLER_CLK_SOURCE_UART1_0

Offset: 178h | Read/Write: R/W | Reset: 0b11

Bit	Reset	Description
31:30	CLK_M	UART1_CLK_SRC: 00 = pllP_out0 01 = pllC_out0 10 = pllM_out0 11 = clk_m 0 = PLLP_OUT0 1 = PLLC_OUT0 2 = PLLM_OUT0 3 = CLK_M

5.4.75 CLK_RST_CONTROLLER_CLK_SOURCE_UART2_0

Offset: 17ch | Read/Write: R/W | Reset: 0b11

Bit	Reset	Description
31:30	CLK_M	UART2_CLK_SRC: 00 = pllP_out0 01 = pllC_out0 10 = pllM_out0 11 = clk_m 0 = PLLP_OUT0 1 = PLLC_OUT0 2 = PLLM_OUT0 3 = CLK_M

5.4.76 CLK_RST_CONTROLLER_CLK_SOURCE_HOST1X_0

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

Bit	Reset	Description
31:30	PLLM_OUT0	HOST1X_CLK_SRC: 00 = pllM_out0 01 = pllC_out0 10 = pllP_out0 11 = pllA_out0 0 = PLLM_OUT0 1 = PLLC_OUT0 2 = PLLP_OUT0 3 = PLLA_OUT0
15:8	0x0	HOST1X_IDLE_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x) if all 0's, this idle divisor field will not be used. For non-zero values, when host1x is idle, this field will be used instead of HOST1X_CLK_DIVISOR.
7:0	0x0	HOST1X_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

5.4.77 CLK_RST_CONTROLLER_CLK_SOURCE_TVO_0

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

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

5.4.78 CLK_RST_CONTROLLER_CLK_SOURCE_HDMI_0

Offset: 18ch | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxxxxxxxxxxx00000000

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

5.4.79 CLK_RST_CONTROLLER_CLK_SOURCE_TVDAC_0

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

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

5.4.80 CLK_RST_CONTROLLER_CLK_SOURCE_I2C2_0

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

Bit	Reset	Description
31:30	CLK_M	I2C2_CLK_SRC: 00 = pllP_out0 01 = pllC_out0 10 = pllM_out0 11 = clk_m 0 = PLLP_OUT0 1 = PLLC_OUT0 2 = PLLM_OUT0 3 = CLK_M
15:0	0x0	I2C2_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 1.0x)

5.4.81 CLK_RST_CONTROLLER_CLK_SOURCE_EMC_0

Offset: 19ch | Read/Write: R/W | Reset: 0b110xxx00xxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	EMC_2X_CLK_SRC: 00 = pllM_out0 01 = pllC_out0 10 = pllP_out0 11 = clk_m 0 = PLLM_OUT0 1 = PLLC_OUT0 2 = PLLP_OUT0 3 = CLK_M
29	0x0	USE_PLLM_UD: 1 = use un-divided PllM_out0 as clock source.
25	DISABLE	EMC_2X_CLK_ENB: 1 = enable EMC 2X clock. 0 = DISABLE 1 = ENABLE
24	DISABLE	EMC_1X_CLK_ENB: 1 = enable EMC 1X clock. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
7:0	0x0	EMC_2X_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

5.4.82 CLK_RST_CONTROLLER_CLK_SOURCE_UART3_0

Offset: 1a0h | Read/Write: R/W | Reset: 0b11

Bit	Reset	Description
31:30	CLK_M	UART3_CLK_SRC: 00 = pllP_out0 01 = pllC_out0 10 = pllM_out0 11 = CLK_M 0 = PLLP_OUT0 1 = PLLC_OUT0 2 = PLLM_OUT0 3 = CLK_M

5.4.83 CLK_RST_CONTROLLER_CLK_SOURCE_VI_SENSOR_0

Offset: 1a8h | Read/Write: R/W | Reset: 0b00xxxxxxxxxxxxxxxxxxxxxxx00000000

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

5.4.84 CLK_RST_CONTROLLER_CLK_SOURCE_SPI4_0

Offset: 1b4h | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	SPI4_CLK_SRC: 00 = pllP_out0 01 = pllC_out0 10 = pllM_out0 11 = clk_m 0 = PLLP_OUT0 1 = PLLC_OUT0 2 = PLLM_OUT0 3 = CLK_M
7:0	0x0	SPI4_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

5.4.85 CLK_RST_CONTROLLER_CLK_SOURCE_I2C3_0

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

Bit	Reset	Description
31:30	CLK_M	I2C3_CLK_SRC: 00 = pllP_out0 01 = pllC_out0 10 = pllM_out0 11 = clk_m 0 = PLLP_OUT0 1 = PLLC_OUT0 2 = PLLM_OUT0 3 = CLK_M
15:0	0x0	I2C3_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 1.0x)

5.4.86 CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC3_0

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

Bit	Reset	Description
31:30	CLK_M	SDMMC3_CLK_SRC: 00 = pllP_out0 01 = pllC_out0 10 = pllM_out0 11 = clk_m 0 = PLLP_OUT0 1 = PLLC_OUT0 2 = PLLM_OUT0 3 = CLK_M
7:0	0x0	SDMMC3_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

5.4.87 CLK_RST_CONTROLLER_CLK_SOURCE_UART4_0

Offset: 1c0h | Read/Write: R/W | Reset: 0b11

Bit	Reset	Description
31:30	CLK_M	UART4_CLK_SRC: 00 = pllP_out0 01 = pllC_out0 10 = pllM_out0 11 = clk_m 0 = PLLP_OUT0 1 = PLLC_OUT0 2 = PLLM_OUT0 3 = CLK_M

5.4.88 CLK_RST_CONTROLLER_CLK_SOURCE_UART5_0

Offset: 1c4h | Read/Write: R/W | Reset: 0b11

Bit	Reset	Description
31:30	CLK_M	UART5_CLK_SRC: 00 = pllP_out0 01 = pllC_out0 10 = pllM_out0 11 = clk_m 0 = PLLP_OUT0 1 = PLLC_OUT0 2 = PLLM_OUT0 3 = CLK_M

5.4.89 CLK_RST_CONTROLLER_CLK_SOURCE_VDE_0

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

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

5.4.90 CLK_RST_CONTROLLER_CLK_SOURCE_OWR_0

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

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

5.4.91 CLK_RST_CONTROLLER_CLK_SOURCE_NOR_0

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

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

5.4.92 CLK_RST_CONTROLLER_CLK_SOURCE_CSITE_0

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

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

5.4.93 CLK_RST_CONTROLLER_CLK_SOURCE_OSC_0

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

Bit	Reset	Description
28	EXT_OSC	OSC_CLK_SRC: 0 = external oscillator 1 = internal PLL_S 0 = EXT_OSC 1 = INT_PLLS_OUT

5.4.94 CLK_RST_CONTROLLER_RST_DEV_L_SET_0

The RST_DEV_(L,H,U)_(SET,CLR) and CLK_ENB_(L,H,U)_(SET,CLR) registers are provided as an alternate method of programming the same registers found in RST_DEVICES_(L,H,U) and CLK_OUT_ENB_(L,H,U) registers, respectively. Therefore, using either methods will change the same underlying peripherals reset, and clock enable control for write. As for read, again, user can use either method to retrieve the reset/clock-enable state of each peripherals.

Offset: 300h | Read/Write: R/W | Reset: 0b0x11111111111111111111011001001

Bit	Reset	Description
31	0x0	SET_CACHE2_RST: set reset COP cache controller.
29	0x1	SET_VCP_RST: set reset vector co-processor.
28	0x1	SET_HOST1X_RST: set reset HOST1X.
27	0x1	SET_DISP1_RST: set reset DISP1 controller.
26	0x1	SET_DISP2_RST: set reset DISP2 controller.
25	0x1	SET_IDE_RST: set reset IDE controller.
24	0x1	SET_3D_RST: set reset 3D controller.
23	0x1	SET_ISP_RST: set reset ISP controller.
22	0x1	SET_USBD_RST: set reset USB controller
21	0x1	SET_2D_RST: set reset 2D graphics engine controller.
20	0x1	SET_VI_RST: set reset VI controller.
19	0x1	SET_EPP_RST: set reset EPP controller.
18	0x1	SET_I2S2_RST: set reset I2S 2 Controller
17	0x1	SET_PWM_RST: set reset Pulse Width Modulator
16	0x1	SET_TWC_RST: set reset Three Wire Controller
15	0x1	SET_SDMMC4_RST: set reset SDMMC4 controller.
14	0x1	SET_SDMMC1_RST: set reset SDMMC1 controller.
13	0x1	SET_NDFLASH_RST: set reset NAND flash controller.
12	0x1	SET_I2C1_RST: set reset I2C1 Controller
11	0x1	SET_I2S1_RST: set reset I2S 1 Controller
10	0x1	SET_SPDIF_RST: set reset SPDIF Controller
9	0x1	SET_SDMMC2_RST: set reset SDMMC2 Controller
8	0x0	SET_GPIO_RST: set reset GPIO Controller
7	0x1	SET_UART2_RST: set reset UART2/VFIR Controller
6	0x1	SET_UART1_RST: set reset UART1 Controller
5	0x0	SET_TMR_RST: set reset Timer Controller
3	0x1	SET_AC97_RST: set reset AC97 Controller

Bit	Reset	Description
2	0x0	SET_TRIG_SYS_RST: Write 1 to pulse System Reset Signal.
1	0x0	SET_COP_RST: set reset COP.
0	0x1	SET_CPU_RST: set reset CPU.

5.4.95 CLK_RST_CONTROLLER_RST_DEV_L_CLR_0

Offset: 304h | Read/Write: R/W | Reset: 0b0x11111111111111111111011001x01

Bit	Reset	Description
31	0x0	CLR_CACHE2_RST: clear reset COP cache controller.
29	0x1	CLR_VCP_RST: clear reset vector co-processor.
28	0x1	CLR_HOST1X_RST: clear reset HOST1X.
27	0x1	CLR_DISP1_RST: clear reset DISP1 controller.
26	0x1	CLR_DISP2_RST: clear reset DISP2 controller.
25	0x1	CLR_IDE_RST: clear reset IDE controller.
24	0x1	CLR_3D_RST: clear reset 3D controller.
23	0x1	CLR_ISP_RST: clear reset ISP controller.
22	0x1	CLR_USBD_RST: clear reset USB controller
21	0x1	CLR_2D_RST: clear reset 2D graphics engine controller.
20	0x1	CLR_VI_RST: clear reset VI controller.
19	0x1	CLR_EPP_RST: clear reset EPP controller.
18	0x1	CLR_I2S2_RST: clear reset I2S 2 Controller
17	0x1	CLR_PWM_RST: clear reset Pulse Width Modulator
16	0x1	CLR_TWC_RST: clear reset Three Wire Controller
15	0x1	CLR_SDMMC4_RST: clear reset SDMMC4 controller.
14	0x1	CLR_SDMMC1_RST: clear reset SDMMC1 controller.
13	0x1	CLR_NDFLASH_RST: clear reset NAND flash controller.
12	0x1	CLR_I2C1_RST: clear reset I2C1 Controller
11	0x1	CLR_I2S1_RST: clear reset I2S 1 Controller
10	0x1	CLR_SPDIF_RST: clear reset SPDIF Controller
9	0x1	CLR_SDMMC2_RST: clear reset SDMMC2 Controller
8	0x0	CLR_GPIO_RST: clear reset GPIO Controller
7	0x1	CLR_UART2_RST: clear reset UART2/VFIR Controller
6	0x1	CLR_UART1_RST: clear reset UART1 Controller

Bit	Reset	Description
5	0x0	CLR_TMR_RST: clear reset Timer Controller
3	0x1	CLR_AC97_RST: clear reset AC97 Controller
1	0x0	CLR_COP_RST: clear reset COP.
0	0x1	CLR_CPU_RST: clear reset CPU.

5.4.96 CLK_RST_CONTROLLER_RST_DEV_H_SET_0

Offset: 308h | Read/Write: R/W | Reset: 0b1111111x11111111111110110111x111

Bit	Reset	Description
31	0x1	SET_BSEV_RST: set reset BSEV controller.
30	0x1	SET_BSEA_RST: set reset BSEA controller.
29	0x1	SET_VDE_RST: set reset VDE controller.
28	0x1	SET_MPE_RST: set reset MPE controller.
27	0x1	SET_USB3_RST: set reset USB3 controller.
26	0x1	SET_USB2_RST: set reset USB2 controller.
25	0x1	SET_EMC_RST: set reset EMC controller.
23	0x1	SET_UART3_RST: set reset UART3 Controller
22	0x1	SET_I2C2_RST: set reset I2C 2 controller.
21	0x1	SET_TVDAC_RST: set reset TVDAC controller.
20	0x1	SET_CSI_RST: set reset CSI controller.
19	0x1	SET_HDMI_RST: set reset HDMI
18	0x1	SET_MIPI_RST: set reset MIPI base-band controller.
17	0x1	SET_TVO_RST: set reset TVO/CVE controller
16	0x1	SET_DSI_RST: set reset DSI controller
15	0x1	SET_DVC_I2C_RST: set reset DVC-I2C Controller
14	0x1	SET_SBC3_RST: set reset SBC 3 (SPI 3) Controller.
13	0x1	SET_XIO_RST: set reset XIO controller.
12	0x1	SET_SBC2_RST: set reset SBC 2 (SPI 2) Controller.
11	0x1	SET_SPI1_RST: set reset SPI 1 Controller
10	0x0	SET_SNOR_RST: set reset NOR Flash Controller.
9	0x1	SET_SBC1_RST: set reset SBC 1 (SPI 1) Controller.
8	0x1	SET_KFUSE_RST: set reset KFuse controller.
5	0x1	SET_STAT_MON_RST: set reset statistic monitor

Bit	Reset	Description
2	0x1	SET_APBDMA_RST: set reset APB-DMA.
1	0x1	SET_AHBDMA_RST: set reset AHB-DMA.
0	0x1	SET_MEM_RST: set reset MC.

5.4.97 CLK_RST_CONTROLLER_RST_DEV_H_CLR_0

Offset: 30ch | Read/Write: R/W | Reset: 0b1111111x11111111111110110111x111

Bit	Reset	Description
31	0x1	CLR_BSEV_RST: clear reset BSEV controller.
30	0x1	CLR_BSEA_RST: clear reset BSEA controller.
29	0x1	CLR_VDE_RST: clear reset VDE controller.
28	0x1	CLR_MPE_RST: clear reset MPE controller.
27	0x1	CLR_USB3_RST: clear reset USB3 controller.
26	0x1	CLR_USB2_RST: clear reset USB2 controller.
25	0x1	CLR_EMC_RST: clear reset EMC controller.
23	0x1	CLR_UART3_RST: clear reset UART3 Controller
22	0x1	CLR_I2C2_RST: clear reset I2C 2 controller.
21	0x1	CLR_TVDAC_RST: clear reset TVDAC controller.
20	0x1	CLR_CSI_RST: clear reset CSI controller.
19	0x1	CLR_HDMI_RST: clear reset HDMI
18	0x1	CLR_MIPI_RST: clear reset MIPI base-band controller.
17	0x1	CLR_TVO_RST: clear reset TVO/CVE controller
16	0x1	CLR_DSI_RST: clear reset DSI controller
15	0x1	CLR_DVC_I2C_RST: clear reset DVC-I2C Controller
14	0x1	CLR_SBC3_RST: clear reset SBC 3 (SPI 3) Controller.
13	0x1	CLR_XIO_RST: clear reset XIO controller.
12	0x1	CLR_SBC2_RST: clear reset SBC 2 (SPI 2)Controller.
11	0x1	CLR_SPI1_RST: clear reset SPI 1 Controller
10	0x0	CLR_SNOR_RST: clear reset NOR Flash Controller.
9	0x1	CLR_SBC1_RST: clear reset SBC 1 (SPI 1) Controller.
8	0x1	CLR_KFUSE_RST: clear reset KFuse controller.
5	0x1	CLR_STAT_MON_RST: clear reset statistic monitor
2	0x1	CLR_APBDMA_RST: clear reset APB-DMA.

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

5.4.98 CLK_RST_CONTROLLER_RST_DEV_U_SET_0

Offset: 310h | Read/Write: R/W | Reset: 0b010111111111

Bit	Reset	Description
11	0x0	SET_AVPUCQ_RST: set reset AVPUCQ logic.
10	0x1	SET_PCIEXCLK_RST: set reset PCIEXCLK logic.
9	0x0	SET_CSITE_RST: set reset CSITE controller.
8	0x1	SET_AFI_RST: set reset AFI controller.
7	0x1	SET_OWR_RST: set reset OWR controller.
6	0x1	SET_PCIE_RST: set reset PCIE controller.
5	0x1	SET_SDMMC3_RST: set reset SDMMC3 controller.
4	0x1	SET_SBC4_RST: set reset SBC4 (SPI 4) controller.
3	0x1	SET_I2C3_RST: set reset I2C3 controller.
2	0x1	SET_UART5_RST: set reset UART5 controller.
1	0x1	SET_UART4_RST: set reset UART4 controller.
0	0x1	SET_SPEEDO_RST: set reset SPEEDO controller.

5.4.99 CLK_RST_CONTROLLER_RST_DEV_U_CLR_0

Offset: 314h | Read/Write: R/W | Reset: 0b010111111111

Bit	Reset	Description
11	0x0	CLR_AVPUCQ_RST: clear reset AVPUCQ logic.
10	0x1	CLR_PCIEXCLK_RST: clear reset PCIEXCLK logic.
9	0x0	CLR_CSITE_RST: clear reset CSITE controller.
8	0x1	CLR_AFI_RST: clear reset AFI controller.
7	0x1	CLR_OWR_RST: clear reset OWR controller.
6	0x1	CLR_PCIE_RST: clear reset PCIE controller.
5	0x1	CLR_SDMMC3_RST: clear reset SDMMC3 controller.
4	0x1	CLR_SBC4_RST: clear reset SBC4 (SPI 4) controller.
3	0x1	CLR_I2C3_RST: clear reset I2C3 controller.
2	0x1	CLR_UART5_RST: clear reset UART5 controller.
1	0x1	CLR_UART4_RST: clear reset UART4 controller.

Bit	Reset	Description
0	0x1	CLR_SPEEDO_RST: clear reset SPEEDO controller.

5.4.100 CLK_RST_CONTROLLER_CLK_ENB_L_SET_0

Offset: 320h | Read/Write: R/W | Reset: 0b1x00000000000000000000100110xx0

Bit	Reset	Description
31	0x1	SET_CLK_ENB_CACHE2: set enable clock to COP cache controller.
29	0x0	SET_CLK_ENB_VCP: set enable clock to vector co-processor.
28	0x0	SET_CLK_ENB_HOST1X: set enable clock to HOST1X.
27	0x0	SET_CLK_ENB_DISP1: set enable clock to DISP1 controller.
26	0x0	SET_CLK_ENB_DISP2: set enable clock to DISP2 controller.
25	0x0	SET_CLK_ENB_IDE: set enable clock to IDE controller
24	0x0	SET_CLK_ENB_3D: set enable clock to 3D controller.
23	0x0	SET_CLK_ENB_ISP: set enable clock to ISP controller.
22	0x0	SET_CLK_ENB_USBD: set enable clock to USB controller
21	0x0	SET_CLK_ENB_2D: set enable clock to 2D graphics engine.
20	0x0	SET_CLK_ENB_VI: set enable clock to VI controller.
19	0x0	SET_CLK_ENB_EPP: set enable clock to EPP controller.
18	0x0	SET_CLK_ENB_I2S2: set enable clock to I2S 2 controller
17	0x0	SET_CLK_ENB_PWM: set enable clock to PWM (Pulse Width Modulator)
16	0x0	SET_CLK_ENB_TWC: set enable clock to 3-Wire Interface Controller
15	0x0	SET_CLK_ENB_SDMMC4: set enable clock to SDMMC4 controller.
14	0x0	SET_CLK_ENB_SDMMC1: set enable clock to SDMMC1 controller.
13	0x0	SET_CLK_ENB_NDFLASH: set enable clock to NAND flash controller.
12	0x0	SET_CLK_ENB_I2C1: set enable clock to I2C1 Controller
11	0x0	SET_CLK_ENB_I2S1: set enable clock to I2S1 Controller
10	0x0	SET_CLK_ENB_SPDIF: set enable clock to SPDIF Controller
9	0x0	SET_CLK_ENB_SDMMC2: set enable clock to SDMMC2 controller.
8	0x1	SET_CLK_ENB_GPIO: set enable clock to GPIO Controller
7	0x0	SET_CLK_ENB_UART2: set enable clock to UART2/VFIR Controller
6	0x0	SET_CLK_ENB_UART1: set enable clock to UART1 Controller
5	0x1	SET_CLK_ENB_TMR: set enable clock to Timer Controller
4	0x1	SET_CLK_ENB_RTC: set enable clock to RTC Controller

Bit	Reset	Description
3	0x0	SET_CLK_ENB_AC97: set enable clock to AC97 Controller
0	0x0	SET_CLK_ENB_CPU: set enable clock to CPU.

5.4.101 CLK_RST_CONTROLLER_CLK_ENB_L_CLR_0

Offset: 324h | Read/Write: R/W | Reset: 0b1x00000000000000000000100110xx0

Bit	Reset	Description
31	0x1	CLR_CLK_ENB_CACHE2: clear enable clock to COP cache controller.
29	0x0	CLR_CLK_ENB_VCP: clear enable clock to vector co-processor.
28	0x0	CLR_CLK_ENB_HOST1X: clear enable clock to HOST1X.
27	0x0	CLR_CLK_ENB_DISP1: clear enable clock to DISP1 controller.
26	0x0	CLR_CLK_ENB_DISP2: clear enable clock to DISP2 controller.
25	0x0	CLR_CLK_ENB_IDE: clear enable clock to IDE controller
24	0x0	CLR_CLK_ENB_3D: clear enable clock to 3D controller.
23	0x0	CLR_CLK_ENB_ISP: clear enable clock to ISP controller.
22	0x0	CLR_CLK_ENB_USBD: clear enable clock to USB controller
21	0x0	CLR_CLK_ENB_2D: clear enable clock to 2D graphics engine.
20	0x0	CLR_CLK_ENB_VI: clear enable clock to VI controller.
19	0x0	CLR_CLK_ENB_EPP: clear enable clock to EPP controller.
18	0x0	CLR_CLK_ENB_I2S2: clear enable clock to I2S 2 controller
17	0x0	CLR_CLK_ENB_PWM: clear enable clock to PWM (Pulse Width Modulator)
16	0x0	CLR_CLK_ENB_TWC: clear enable clock to 3-Wire Interface Controller
15	0x0	CLR_CLK_ENB_SDMMC4: clear enable clock to SDMMC4 controller.
14	0x0	CLR_CLK_ENB_SDMMC1: clear enable clock to SDMMC1 controller.
13	0x0	CLR_CLK_ENB_NDFLASH: clear enable clock to NAND flash controller.
12	0x0	CLR_CLK_ENB_I2C1: clear enable clock to I2C1 Controller
11	0x0	CLR_CLK_ENB_I2S1: clear enable clock to I2S1 Controller
10	0x0	CLR_CLK_ENB_SPDIF: clear enable clock to SPDIF Controller
9	0x0	CLR_CLK_ENB_SDMMC2: clear enable clock to SDMMC2 controller.
8	0x1	CLR_CLK_ENB_GPIO: clear enable clock to GPIO Controller
7	0x0	CLR_CLK_ENB_UART2: clear enable clock to UART2/VFIR Controller
6	0x0	CLR_CLK_ENB_UART1: clear enable clock to UART1 Controller
5	0x1	CLR_CLK_ENB_TMR: clear enable clock to Timer Controller

Bit	Reset	Description
4	0x1	CLR_CLK_ENB_RTC: clear enable clock to RTC Controller
3	0x0	CLR_CLK_ENB_AC97: clear enable clock to AC97 Controller
0	0x0	CLR_CLK_ENB_CPU: clear enable clock to CPU.

5.4.102 CLK_RST_CONTROLLER_CLK_ENB_H_SET_0

Offset: 328h | Read/Write: R/W | Reset: 0b0000000x00000000000001000000x000

Bit	Reset	Description
31	0x0	SET_CLK_ENB_BSEV: set enable clock to BSEV Controller.
30	0x0	SET_CLK_ENB_BSEA: set enable clock to BSEA Controller
29	0x0	SET_CLK_ENB_VDE: set enable clock to VDE Controller
28	0x0	SET_CLK_ENB_MPE: set enable clock to MPE controller.
27	0x0	SET_CLK_ENB_USB3: set enable clock to USB3 controller.
26	0x0	SET_CLK_ENB_USB2: set enable clock to USB2 controller.
25	0x0	SET_CLK_ENB_EMC: set enable clock to EMC controller.
23	0x0	SET_CLK_ENB_UART3: set enable clock to UART3 Controller
22	0x0	SET_CLK_ENB_I2C2: set enable clock to I2C2 controller.
21	0x0	SET_CLK_ENB_TVDAC: set enable clock to TVDAC controller.
20	0x0	SET_CLK_ENB_CSI: set enable clock to CSI controller
19	0x0	SET_CLK_ENB_HDMI: set enable clock to HDMI
18	0x0	SET_CLK_ENB_MIPI: set enable clock to MIPI base-band controller
17	0x0	SET_CLK_ENB_TVO: set enable clock to TVO/CVE controller
16	0x0	SET_CLK_ENB_DSI: set enable clock to DSI controller
15	0x0	SET_CLK_ENB_DVC_I2C: set enable clock to DVC-I2C Controller
14	0x0	SET_CLK_ENB_SBC3: set enable clock to SBC 3 (SPI 3) Controller.
13	0x0	SET_CLK_ENB_XIO: set enable clock to XIO Controller.
12	0x0	SET_CLK_ENB_SBC2: set enable clock to SBC 2 (SPI 2) Controller.
11	0x0	SET_CLK_ENB_SPI1: set enable clock to SPI 1 Controller
10	0x1	SET_CLK_ENB_SNOR: set enable clock to NOR Flash Controller.
9	0x0	SET_CLK_ENB_SBC1: set enable clock to SBC 1 (SPI 1) Controller.
8	0x0	SET_CLK_ENB_KFUSE: set enable clock to KFUSE controller.
7	0x0	SET_CLK_ENB_FUSE: set enable clock to FUSE controller.
6	0x0	SET_CLK_ENB_PMC: set enable clock to PMC controller.

Bit	Reset	Description
5	0x0	SET_CLK_ENB_STAT_MON: set enable clock to statistic monitor.
4	0x0	SET_CLK_ENB_KBC: set enable clock to keyboard controller.
2	0x0	SET_CLK_ENB_APBDMA: set enable clock to APB-DMA.
1	0x0	SET_CLK_ENB_AHBDMA: set enable clock to AHB-DMA.
0	0x0	SET_CLK_ENB_MEM: set enable clock to MC/EMC.

5.4.103 CLK_RST_CONTROLLER_CLK_ENB_H_CLR_0

Offset: 32ch | Read/Write: R/W | Reset: 0b0000000x00000000000001000000x000

Bit	Reset	Description
31	0x0	CLR_CLK_ENB_BSEV: clear enable clock to BSEV Controller.
30	0x0	CLR_CLK_ENB_BSEA: clear enable clock to BSEA Controller
29	0x0	CLR_CLK_ENB_VDE: clear enable clock to VDE Controller
28	0x0	CLR_CLK_ENB_MPE: clear enable clock to MPE controller.
27	0x0	CLR_CLK_ENB_USB3: clear enable clock to USB3 controller.
26	0x0	CLR_CLK_ENB_USB2: clear enable clock to USB2 controller.
25	0x0	CLR_CLK_ENB_EMC: clear enable clock to EMC controller.
23	0x0	CLR_CLK_ENB_UART3: clear enable clock to UART3 Controller
22	0x0	CLR_CLK_ENB_I2C2: clear enable clock to I2C2 controller.
21	0x0	CLR_CLK_ENB_TVDAC: clear enable clock to TVDAC controller.
20	0x0	CLR_CLK_ENB_CSI: clear enable clock to CSI controller
19	0x0	CLR_CLK_ENB_HDMI: clear enable clock to HDMI
18	0x0	CLR_CLK_ENB_MIPI: clear enable clock to MIPI base-band controller
17	0x0	CLR_CLK_ENB_TVO: clear enable clock to TVO/CVE controller
16	0x0	CLR_CLK_ENB_DSI: clear enable clock to DSI controller
15	0x0	CLR_CLK_ENB_DVC_I2C: clear enable clock to DVC-I2C Controller
14	0x0	CLR_CLK_ENB_SBC3: clear enable clock to SBC 3 (SPI 3) Controller.
13	0x0	CLR_CLK_ENB_XIO: clear enable clock to XIO Controller.
12	0x0	CLR_CLK_ENB_SBC2: clear enable clock to SBC 2 (SPI 2) Controller.
11	0x0	CLR_CLK_ENB_SPI1: clear enable clock to SPI 1 Controller
10	0x1	CLR_CLK_ENB_SNOR: clear enable clock to NOR Flash Controller.
9	0x0	CLR_CLK_ENB_SBC1: clear enable clock to SBC 1 (SPI 1) Controller.
8	0x0	CLR_CLK_ENB_KFUSE: clear enable clock to KFUSE controller.

Bit	Reset	Description
7	0x0	CLR_CLK_ENB_FUSE: clear enable clock to FUSE controller.
6	0x0	CLR_CLK_ENB_PMC: clear enable clock to PMC controller.
5	0x0	CLR_CLK_ENB_STAT_MON: clear enable clock to statistic monitor.
4	0x0	CLR_CLK_ENB_KBC: clear enable clock to keyboard controller.
2	0x0	CLR_CLK_ENB_APBDMA: clear enable clock to APB-DMA.
1	0x0	CLR_CLK_ENB_AHBDMA: clear enable clock to AHB-DMA.
0	0x0	CLR_CLK_ENB_MEM: clear enable clock to MC/EMC.

5.4.104 CLK_RST_CONTROLLER_CLK_ENB_U_SET_0

Offset: 330h | Read/Write: R/W | Reset: 0b000x1111111xxxxxxx01x1000000000

Bit	Reset	Description
30	0x0	SET_CLK_ENB_DEV1_OUT: set enable clock to DEV1 pad.
29	0x0	SET_CLK_ENB_DEV2_OUT: set enable clock to DEV2 pad.
28	0x0	SET_CLK_ENB_SUS_OUT: set enable clock to SUS pad.
26	0x1	SET_CLK_M_DOUBLER_ENB: set enable CLK_M clk doubler.
25	0x1	SET_SYNC_CLK_DOUBLER_ENB: set enable audio sync clk doubler.
24	0x1	SET_CLK_ENB_CRAM2: set enable COP cache ram clk.
23	0x1	SET_CLK_ENB_IRAMD: set enable IRAMD clk.
22	0x1	SET_CLK_ENB_IRAMC: set enable IRAMC clk.
21	0x1	SET_CLK_ENB_IRAMB: set enable IRAMB clk.
20	0x1	SET_CLK_ENB_IRAMA: set enable IRAMB clk.
11	0x1	SET_CLK_ENB_AVPUCQ: set enable clock to AVPUCQ.
9	0x1	SET_CLK_ENB_CSITE: set enable clock to CSITE.
8	0x0	SET_CLK_ENB_AFI: set enable clock to AFI.
7	0x0	SET_CLK_ENB_OWR: set enable clock to OWR.
6	0x0	SET_CLK_ENB_PCIE: set enable clock to PCIE.
5	0x0	SET_CLK_ENB_SDMMC3: set enable clock to SDMMC3.
4	0x0	SET_CLK_ENB_SBC4: set enable clock to SBC4 (SPI 4).
3	0x0	SET_CLK_ENB_I2C3: set enable clock to I2C3.
2	0x0	SET_CLK_ENB_UART5: set enable clock to UART5.
1	0x0	SET_CLK_ENB_UART4: set enable clock to UART4.
0	0x0	SET_CLK_ENB_SPEEDO: set enable clock to SPEEDO.

5.4.105 CLK_RST_CONTROLLER_CLK_ENB_U_CLR_0

Offset: 334h | Read/Write: R/W | Reset: 0b000x1111111xxxxxxx01x1000000000

Bit	Reset	Description
30	0x0	CLR_CLK_ENB_DEV1_OUT: clear enable clock to DEV1 pad.
29	0x0	CLR_CLK_ENB_DEV2_OUT: clear enable clock to DEV2 pad.
28	0x0	CLR_CLK_ENB_SUS_OUT: clear enable clock to SUS pad.
26	0x1	CLR_CLK_M_DOUBLER_ENB: clear enable CLK_M clk doubler.
25	0x1	CLR_SYNC_CLK_DOUBLER_ENB: clear enable audio sync clk doubler.
24	0x1	CLR_CLK_ENB_CRAM2: clear enable COP cache ram clk.
23	0x1	CLR_CLK_ENB_IRAMD: clear enable IRAMD clk.
22	0x1	CLR_CLK_ENB_IRAMC: clear enable IRAMC clk.
21	0x1	CLR_CLK_ENB_IRAMB: clear enable IRAMB clk.
20	0x1	CLR_CLK_ENB_IRAMA: clear enable IRAMB clk.
11	0x1	CLR_CLK_ENB_AVPUCQ: clear enable clock to AVPUCQ.
9	0x1	CLR_CLK_ENB_CSITE: clear enable clock to CSITE.
8	0x0	CLR_CLK_ENB_AFI: clear enable clock to AFI.
7	0x0	CLR_CLK_ENB_OWR: clear enable clock to OWR.
6	0x0	CLR_CLK_ENB_PCIE: clear enable clock to PCIE.
5	0x0	CLR_CLK_ENB_SDMMC3: clear enable clock to SDMMC3.
4	0x0	CLR_CLK_ENB_SBC4: clear enable clock to SBC4 (SPI 4).
3	0x0	CLR_CLK_ENB_I2C3: clear enable clock to I2C3.
2	0x0	CLR_CLK_ENB_UART5: clear enable clock to UART5.
1	0x0	CLR_CLK_ENB_UART4: clear enable clock to UART4.
0	0x0	CLR_CLK_ENB_SPEEDO: clear enable clock to SPEEDO.

5.4.106 CLK_RST_CONTROLLER_RST_CPU_CMPLX_SET_0

Offset: 340h | Read/Write: R/W | Reset: 0b000xxxxxxxxxxxxx10xx10xx10xx10

Bit	Reset	Description
30	0x0	SET_PRESETDBG: 1 = assert nPRESETDBG to the coresight.
29	0x0	SET_SCURESET: 1 = assert nSCURESET to the SCU.
28	0x0	SET_PERIPHRESET: 1 = assert nPERIPHRESET to the CPU.
13	0x1	SET_DBGRESET1: 1 = assert nDBGRESET to CPU1.
12	0x0	SET_DBGRESET0: 1 = assert nDBGRESET to CPU0.

Bit	Reset	Description
9	0x1	SET_WDRESET1: 1 = assert nWDRESET to CPU1.
8	0x0	SET_WDRESET0: 1 = assert nWDRESET to CPU0.
5	0x1	SET_DERESET1: 1 = assert nDERESET to CPU1.
4	0x0	SET_DERESET0: 1 = assert nDERESET to CPU0.
1	0x1	SET_CPURESET1: 1 = assert nCPURESET to CPU1.
0	0x0	SET_CPURESET0: 1 = assert nCPURESET to CPU0.

5.4.107 CLK_RST_CONTROLLER_RST_CPU_CMPLX_CLR_0

Offset: 344h | Read/Write: R/W | Reset: 0b000xxxxxxxxxxxxx10xx10xx10xx10

Bit	Reset	Description
30	0x0	CLR_PRESETDBG: 1 = deassert nPRESETDBG to the coresight.
29	0x0	CLR_SCURESET: 1 = deassert nSCURESET to the SCU.
28	0x0	CLR_PERIPHRESET: 1 = deassert nPERIPHRESET to the CPU's interrupt/timer.
13	0x1	CLR_DBGRESET1: 1 = deassert nDBGRESET to CPU1.
12	0x0	CLR_DBGRESET0: 1 = deassert nDBGRESET to CPU0.
9	0x1	CLR_WDRESET1: 1 = deassert nWDRESET to CPU1.
8	0x0	CLR_WDRESET0: 1 = deassert nWDRESET to CPU0.
5	0x1	CLR_DERESET1: 1 = deassert nDERESET to CPU1.
4	0x0	CLR_DERESET0: 1 = deassert nDERESET to CPU0.
1	0x1	CLR_CPURESET1: 1 = deassert nCPURESET to CPU1.
0	0x0	CLR_CPURESET0: 1 = deassert nCPURESET to CPU0.

6.0 REAL-TIME CLOCK

The Real-Time Clock (RTC) module maintains seconds and milliseconds counters, and five alarm registers. RTC is in 'always-on' power domain, allowing for the counters to run and alarms to trigger when the system is in low-power state. If configured, interrupts triggered by RTC can cause the system to wake up from a low-power state.

Features

- 10-bit milliseconds counter that runs off of a 32.768 KHz clock source.
- 32-bit second counter that increment for every 1000 milliseconds.
- Alarm feature that triggers an interrupt when specified value matches the millisecond counter.
- Alarm feature that triggers an interrupt when 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 RTC clock keeps running.
- Hardware adjusts drift in clock which can occur due to ppm variations in oscillator output.

6.1 Functional Description

The RTC operates in two clock domains; APB clock domain, and 32 KHz clock domain. RTC continues updating the millisecond and second counters and continues triggering interrupts even when the MAIN partition is powered down. Since the APB clock is disabled when MAIN is powered down, registers are implemented in the 32 KHz clock domain.

All the registers except the BUSY register are implemented in 32 KHz clock domain. Writes are transferred to 32 KHz domain with BUSY.STATUS set as BUSY until the transfer is completed. Reads are shadowed in APB clock domain and return immediately.

At boot, the seconds counter is updated to reflect the current time. Writes to seconds counter can be disabled by writing to CONTROL.DIS_WR_SEC_CNT bit. Alarm registers and countdown alarm registers set interrupt bits when corresponding events occur. Interrupt status, mask, set and source registers reflect the status and also allow setting and clearing of various bits.

Resets

RTC receives an asynchronous reset which is synchronized to APB clock and RTC clock domains.

6.2 RTC Registers

6.2.1 APBDEV_RTC_CONTROL_0

Control register.

Offset: 000h | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
0	0x0	WR_SEC_CNT: When set, writes to SECONDS counter are disabled. can only cleared by resetting the RTC module 0 = DISABLE 1 = ENABLE

6.2.2 APBDEV_RTC_BUSY_0

Busy register.

Offset: 004h | Read/Write: RO | Reset: 0bx

Bit	Reset	Description
0	X	STATUS: This bit is set when a write is initiated on the APB side. It is cleared once the write completes in RTC 32KHz clock domain which could be several thousands of APB clocks. This must be IDLE before a write is initiated. Note that this bit is only for writes. 0 = IDLE 1 = BUSY

6.2.3 APBDEV_RTC_SECONDS_0

Seconds counter register. SECONDS register is copied over to APB side every eight 32KHz clocks (~250uS). Because of this, performing a read immediately after a write might return old value. This covers 49710.26 Days (or) 136.192 Years of 365 days each

Offset: 008h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	SECONDS: seconds counter is incremented for every 1000 milli-seconds

6.2.4 APBDEV_RTC_SHADOW_SECONDS_0

Shadowed seconds counter register. Shadow SECONDS register is updated over to APB side whenever there is a read to milliseconds counter. Since the software cannot read both registers at any given point of time, the Seconds register is snapshotted in this Register. If the software needs to read two registers, then it should read MILLI_SECONDS Register and then this Register. It should not read the SECONDS register, as it contains the updated value.

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

Bit	Reset	Description
31:0	X	SHADOW_SECONDS: A snapshot of the SECONDS counter is taken whenever there is a read to MILLI_SECONDS Register.

6.2.5 APBDEV_RTC_MILLI_SECONDS_0

Milliseconds counter register. Milli SECONDS register is copied over to APB side every eight 32KHz clocks (~250uS). Because of this, performing a read immediately after a write might return old value.

Offset: 010h | Read/Write: RO | Reset: 0bxxxxxxxxxx

Bit	Reset	Description
9:0	X	MILLI_SECONDS: milliseconds counter is incremented using Bresenham algorithm

6.2.6 APBDEV_RTC_SECONDS_ALARM0_0

Seconds alarm0 register. When the value in this register matches the seconds counter, corresponding interrupt status bit is set.

If enabled, the interrupt line is asserted. A value of zero disables the alarm.

Offset: 014h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	SECS_MATCH_VALUE: match value to trigger the alarm

6.2.7 APBDEV_RTC_SECONDS_ALARM1_0

Seconds alarm1 register. When the value in this register matches the seconds counter, corresponding interrupt status bit is set.

If enabled, the interrupt line is asserted. A value of zero disables the alarm.

Offset: 018h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	SECS_MATCH_VALUE: match value to trigger the alarm

6.2.8 APBDEV_RTC_MILLI_SECONDS_ALARM_0

Milliseconds alarm register. When the value in this register matches the milliseconds counter, corresponding interrupt status bit is set.

If enabled, the interrupt line is asserted. A value of zero disables the alarm.

Offset: 01ch | Read/Write: R/W | Reset: 0b0000000000

Bit	Reset	Description
9:0	0x0	MSEC_MATCH_VALUE: milliseconds match value.

6.2.9 APBDEV_RTC_SECONDS_COUNTDOWN_ALARM_0

Countdown alarm registers. If ENABLE_ENABLED, an internal counter is loaded with VALUE and counted down. The interrupt bit corresponding to the countdown_alarm_0 is set after the specified number of seconds have elapsed.

The interrupt bit corresponding to the countdown_alarm_1 is set after the specified number of milliseconds have elapsed. If REPEAT is ENABLED, the countdown operation is performed repeatedly.

Offset: 020h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENABLE: enable bit for the countdown operation. If repeat is not set, this bit is cleared once the internal counters counts down to specified value. 0 = DISABLED 1 = ENABLED
30	0x0	REPEAT: repeat bit for the countdown operation 0 = DISABLED 1 = ENABLED
29:0	0x0	VALUE: number of milliseconds to countdown

6.2.10 APBDEV_RTC_MILLI_SECONDS_COUNTDOWN_ALARM_0

Offset: 024h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENABLE: enable bit for the countdown operation. If repeat is not set, this bit is cleared once the internal counters counts down to specified value. 0 = DISABLED 1 = ENABLED
30	0x0	REPEAT: repeat bit for the countdown operation 0 = DISABLED 1 = ENABLED
29:0	0x0	VALUE: number of milliseconds to countdown

6.2.11 APBDEV_RTC_INTR_MASK_0

Interrupt mask register. This register stores the masks for the interrupts. If a bit is set 1 and the corresponding interrupt condition is satisfied, interrupt line to system interrupt controller is asserted.

Offset: 028h | Read/Write: R/W | Reset: 0b00000

Bit	Reset	Description
4	0x0	MSEC_CDN_ALARM
3	0x0	SEC_CDN_ALARM
2	0x0	MSEC_ALARM
1	0x0	SEC_ALARM1
0	0x0	SEC_ALARM0

6.2.11.1 APBDEV_RTC_INTR_STATUS_0

Interrupt status register. Bits in this register are high after the interrupt condition is satisfied.

A write to this register clears the bits corresponding to the data bits that are high in the write data.

Offset: 02ch | Read/Write: R/W | Reset: 0b00000

Bit	Reset	Description
4	0x0	MSEC_CDN_ALARM
3	0x0	SEC_CDN_ALARM
2	0x0	MSEC_ALARM
1	0x0	SEC_ALARM1
0	0x0	SEC_ALARM0

6.2.11.2 APBDEV_RTC_INTR_SOURCE_0

Interrupt source register. This is a read-only register which returns the AND of interrupt source and interrupt mask registers.

Offset: 030h | Read/Write: R/W | Reset: 0b00000

Bit	Reset	Description
4	0x0	MSEC_CDN_ALARM
3	0x0	SEC_CDN_ALARM
2	0x0	MSEC_ALARM
1	0x0	SEC_ALARM1
0	0x0	SEC_ALARM0

6.2.11.3 APBDEV_RTC_INTR_SET_0

Interrupt set register. This is a write-only register which can be used to set the interrupt status bit.

A write to this register causes the bits in status register to be set if the corresponding bit in write data is 1'b1. Read always returns 'h0.

Offset: 034h | Read/Write: R/W | Reset: 0b00000

Bit	Reset	Description
4	0x0	MSEC_CDN_ALARM
3	0x0	SEC_CDN_ALARM
2	0x0	MSEC_ALARM
1	0x0	SEC_ALARM1
0	0x0	SEC_ALARM0

6.2.11.4 APBDEV_RTC_CORRECTION_FACTOR_0

Correction factor (digital trimming) register. Support is for +/- 500ppm (+/- 50 ppm is maximum)

Offset: 038h | Read/Write: R/W | Reset: 0b0000000000

Bit	Reset	Description
9	0x0	DIRECTION: 0 = DECREMENT 1 = INCREMENT
8:0	0x0	PPM

7.0 TIMERS

The Timer Controller is a 29 bit compound controller. It counts ticks and evaluates/assigns a passage of time to the number of ticks.

The Tegra® 2 Processor has four 29-bit timers that run at 1 MHz clock rate based on the crystal oscillator. Each timer can be programmed to generate single, periodic, or watchdog interrupts.

When the EN bit is set high, the timer loads the timer present trigger value (PTV) into its counter and starts decrementing at a 1 MHz clock rate (1 µsec increment). When the timer present count value (PCR) decrements to zero, it generates a timer interrupt. When the enable periodic interrupt (PER) bit is enabled, the timer generates an interrupt and reloads the counter with the PTV value and starts to decrement again. When the timer consecutively decrements to zero a second time, without the processor reading the first interrupt status register (PCR), it generates a watchdog timer (WDT) interrupt.

A watchdog timer facilitates recovery from system lockup conditions. Either of the two generic timers can be configured as the watchdog timer using the Reset Trigger Source (RST_SOURCE) register. The watchdog timer interrupts the processor when the selected counter expires. Under normal operation, this interrupt is serviced by the processor by reading the status register (timer register).

If the interrupt is not serviced by the processor before the watchdog timer counts down to zero a second time, this is treated as a lockup condition. When this occurs, the watchdog timer generates a reset to the system. Watchdog timer functionality is enabled using bit 5 (WDT.E) of the RST_SOURCE register. Depending on which timer is configured for the watchdog timer function, the processor needs to service the watch dog interrupts by reading the corresponding timer status registers TMR1_PCR and TMR2_PCR. The reset generated by the watchdog timer can be configured to reset the system (entire chip) or the processors individually. This configuration is done using bits [2:0] of the RST_SOURCE register. Software can determine the cause of the reset by checking the status flags (bits [13:8]) of the RST_SOURCE. The USEC_DIVIDEND" and "USEC_DIVISOR" are used to indicate what fraction of 1 microsecond each oscillator clock represents.

Note:	If the xtal oscillator or clock source is disabled during low-power standby mode, the microsecond timers will not function. This also means that interrupts will not occur.

7.1 Functionality

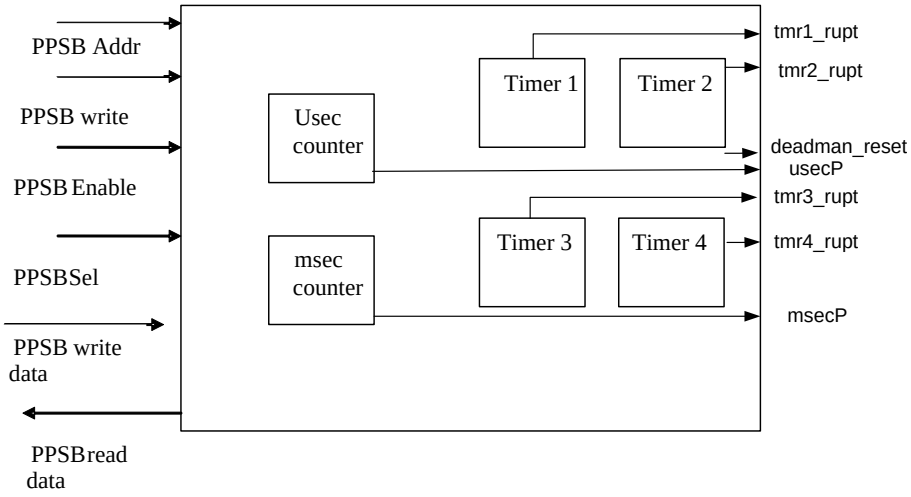
The programmable timer block contains four 29-bit programmable timer counters and a 32-bit time-stamp counter.

Each Timer can be programmed to generate a periodic interrupt or a one-time interrupt. The Timer count decrements by one microsecond when a timer is programmed via a corresponding Enable-bit. The timer generates a timer request whenever the count reaches Zero. If the periodic control bit is set, the timer will reload itself with the count value and decrement again. If the Periodic bit is not set, the timer will generate a single interrupt, clear its enable bit and not reload.

The timer count is 29-bit wide, therefore, it will support a periodic interrupt every 16.384 msec maximum or a minimum of 1 Usec. 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 Decrement counter. The time-stamp counter (32-bit wide) is cleared to Zero on reset, and counts upwards every 1 Usec.

Figure 2. Timer Block Diagram



7.2 Watchdog Timer Programming Guide

The watchdog timer function facilitates recovery from system lockup. The watchdog can be programmed to reset just the CPU, just the AVP, or the entire system.

If only the CPU is reset, then AVP will just continue running as-is. The CPU reset vector should be programmed in advance so that it skips the iROM boot code and jumps directly into its own CPU boot routine (which can be anywhere.)

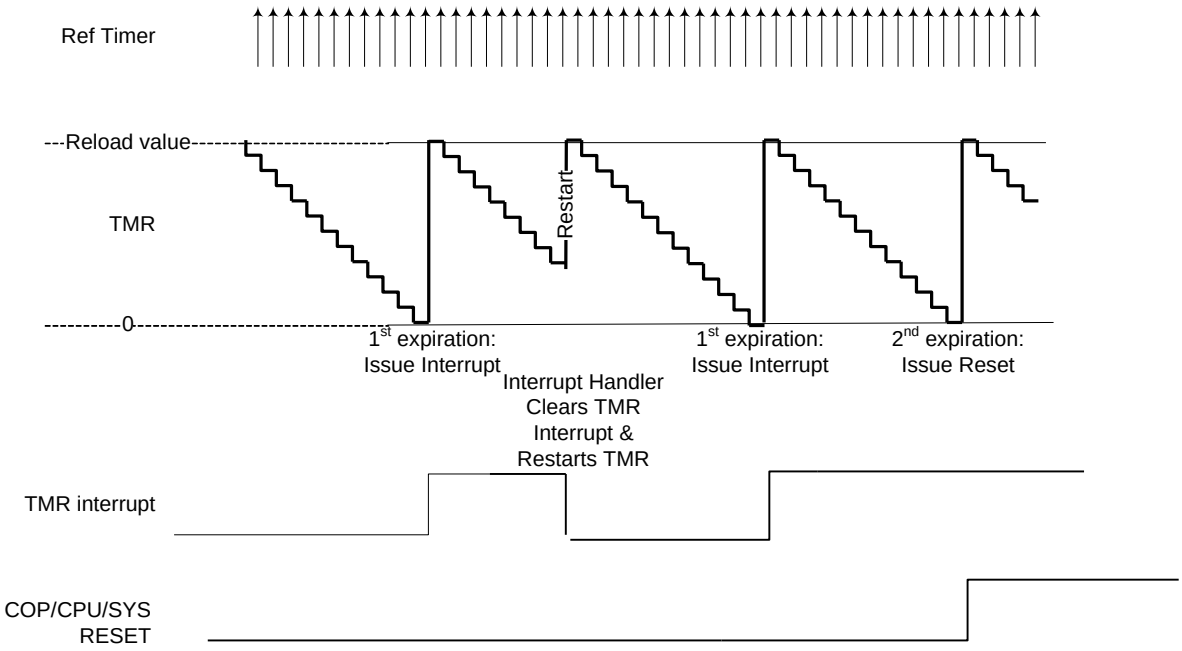
If only the AVP is reset, the same is true. The CPU keeps running as-is. If the AVP reset vector is set to default, it will run the normal boot code. It would be up to the boot loader to decide if it needs to restart the CPU as well.

The watchdog timer can be programmed to max timeout value of 0x3fffffff which is more than 1 billion. At 1µSec/count this is more than one thousand seconds.

It is operated as follows:

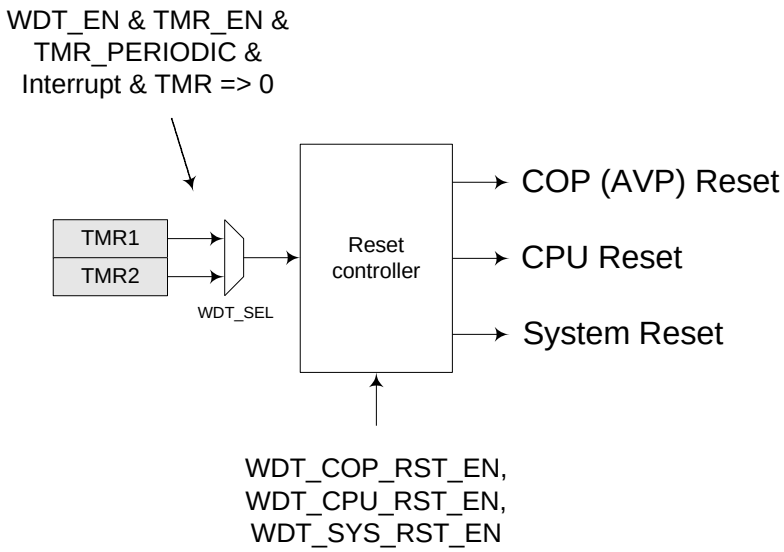
1. Watchdog generates Interrupt based on Timer
    a. TMR1 or TMR2 can be selected as source
    b. Timer should be programmed as Periodic with Interrupt.
    c. CAR.RST_SOURCE.WDT_EN = 1
    d. CAR.RST_SOURCE.WDT_SEL selects TMR2/TMR1
2. Processor must handle interrupt or Reset is issued
    a. First timeout -- interrupt is generated
    b. If interrupt is not handled by next timeout, reset issued
3. Reset can be for AVP, CPU, or full system
    a. CAR.RST_SOURCE.WDT_***_RST_EN = 1
4. Upon reboot, CAR.RST_SOURCE.WDT_*_RST_STA indicates cause of the reset

Figure 3. Watchdog example timing diagram



Although the Timing diagram implies that you need to restart the timer when you service the interrupt, this is not necessary as long as you can always service the interrupt (clear the TMR interrupt) before the next timeout occurs. The reset only happens if a new timeout occurs when the TMR interrupt is still active.

Figure 4. Watchdog Block Diagram



7.3 Timer Registers

7.3.1 TIMER_TMR_PTV_0

Before programming timer:

- Input oscillator frequency must be specified in the TIMERUS_USEC_CFG to support multiple frequencies
- Program the number of 1us timer counts per "tick" in the PTV (Present Trigger Value) register.
- Set the PTV EN (enable bit) to start counting down.
- When the count reaches zero, interrupt is generated.
- To auto-reload the timer counter, set the PTV PER (periodic) bit. Otherwise, leave it clear for a one-shot.

Timer Present Trigger Value (Set) Register

Offset: 000h | Read/Write: R/W | Reset: 0b00x00000000000000000000000000

Bit	Reset	Description
31	0x0	EN: Enable Timer 0 = DISABLE 1 = ENABLE
30	0x0	PER: Enable Periodic Interrupt 0 = DISABLE 1 = ENABLE
28:0	0x0	TMR_PTV: Trigger Value: count trigger value (count length). This is in n+1 scheme. If you program the value n, the count trigger value will actually be n+1.

7.3.2 TIMER_TMR_PCR_0

When interrupt is generated, write "1" to PCR[30] present count Register) to clear the interrupt.

Timer interrupt assignments:

- Timer 1 = primary interrupt controller bit 0
- Timer 2 = primary interrupt controller bit 1
- Timer 3 = secondary interrupt controller bit 9
- Timer 4 = secondary interrupt controller bit 10

Timer Present Count Value (Status) Register

Offset: 004h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
30	RW	0x0	INTR_CLR: 1 = clears the interrupt, 0 = no affect. Write-1-to-Clear
28:0	RO	X	TMR_PCV: Counter value: decrements from PTV.

7.4 Timer USEC CFG

The purpose of USEC_CFG/CNTR_1US registers is to provide a fixed time base (in microseconds) to be used by the rest of the system regardless of the oscillator frequency (i.e. 12 MHz, 13 MHz, 19.2 MHz, 26 MHz, or other frequencies).

7.4.1 USEC_CFG Register

Software should first configure this register by telling what fraction of 1 microsecond each oscillator clock represents. For example, if the oscillator clock is running at 12 MHz, then each oscillator clock represents 1/12 of a micro-second. "USEC_DIVIDEND" and "USEC_DIVISOR" are used to indicate what fraction of 1 microsecond each oscillator clock represents.

Table 24 Oscillator Clock Frequency Indicator for USEC

Oscillator Clock Frequency	Dividend/Divisor	USEC_DIVIDEND/USEC_DIVISOR
12 MHz	1/12	0x00 / 0x0b
13 MHz	1/13	0x00 / 0x0c
19.2 MHz	5/96	0x04 / 0x5f
26 MHz	1/26	0x00 / 0x19

7.5 CNTR_1US Register

This free-running read-only register/counter changes once every micro-second and is used mainly by hardware (can also be used by software). It starts counting from 0 once it's out of system reset and will continue counting forever, unless the oscillator clock is stopped or during a system reset.

(A) Software Use

Although there's no interrupt mechanism for this register, software can read the content of this register multiple times to determine the amount of time that has elapsed.

(B) Hardware Use

- Used by the 4 timers to determine whether the programmed timer value has been reached; these 4 timers can trigger interrupts.
- To provide a periodic USEC pulse to be used by the flow controller to count the programmable number of micro-second before a flow control condition is triggered.
- Used by secure boot logic.

7.5.1 TIMERUS_CNTR_1US_0

Offset: 000h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	x	HIGH_VALUE: Elapsed time in micro-second
15:0	x	LOW_VALUE: Elapsed time in micro-second

7.5.2 TIMERUS_USEC_CFG_0

Offset: 004h | Read/Write: R/W | Reset: 0b0000000000001100 | Default: 0000.0000

Bit	Reset	Description
15:8	0x0	USEC_DIVIDEND: usec dividend.
7:0	0xc	USEC_DIVISOR: usec divisor.

7.5.3 TIMERUS_CNTR_FREEZE_0

The timer disable logic freezes (stops incrementing or decrementing) the timer counters when one or both of the ARM cores enters debug state. The timers remain frozen while either core is in debug state or its freeze enable bit is asserted:

freeze = (cpu_timer_freeze && cpu_debug_state) || (cop_timer_freeze && cop_debug_state)

Offset: 03ch | Read/Write: R/W | Reset: 0b0xx00

Bit	Reset	Description
4	0x0	DBG_FREEZE_COP: 1 = freeze timers when COP is in debug state, 0 = no freeze.
1	0x0	DBG_FREEZE_CPU1: 1 = freeze timers when CPU1 is in debug state, 0 = no freeze.
0	0x0	DBG_FREEZE_CPU0: 1 = freeze timers when CPU0 is in debug state, 0 = no freeze.

8.0 PIN MUXING

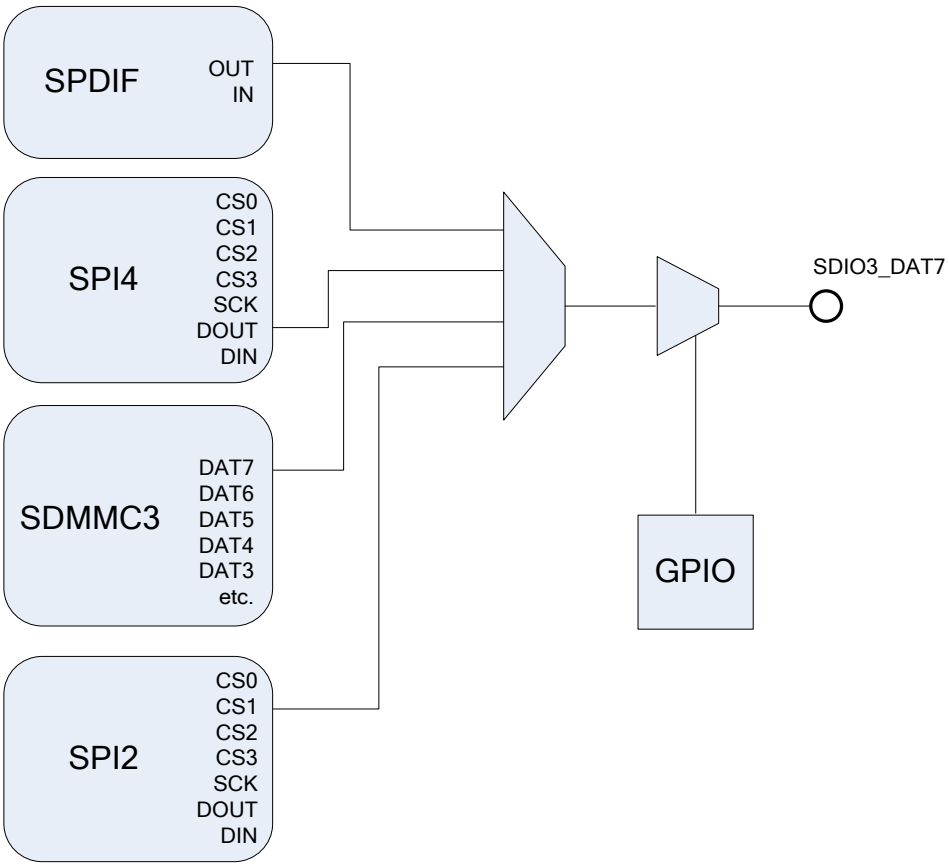
The process of assigning functions to pads is referred to as Pin Muxing, and is available on a large number of the pads. Every pin-mux capable pad on Tegra® 2 Processor may be programmatically assigned from one of a set of up to four functions at run-time. This may be done to reduce the number of pads on the chip.

For example, SDIO3_DAT7 pad may be assigned to one of:

- SPDIF controller's OUT (output) signal
- SPI controller 4's DOUT (data out) signal
- SDMMC controller 3's DAT7 (data bus bit 7) signal
- SPI controller 2's CS1 (chip select 1) signal

Additionally, most pads may be configured to operate as GPIOs, rather than perform the assigned function. A conceptual diagram of Pin Muxing (for the SDIO_DAT7 pad example) is shown in the figure below.

Figure 5. Tegra 2 SDIO3_DAT7 Pad Pin Mux Diagram



8.1 Pad Groups

Pads in the Tegra 2 Processor are internally organized into pad groups, with the pin mux controls applied to the pad group as a whole. For the pads which may be alternately configured as GPIOs, the GPIO assignment is on a per-pad basis, rather than a pad group basis.

The list of pad groups which have pin mux controls and the pads within each pad group can be found in the table below.

Table 25. Tegra 2 Series Pad Groups

Group	Pads
ATA	GMI_CS[7:6]_N GMI_RST_N
ATB	GMI_CS5_N GMI_DPD
ATC	GMI_IORDY GMI_WAIT GMI_ADV_N GMI_CLK GMI_CS[4:2]_N GMI_AD[7:0] GMI_WR_N GMI_OE_N
ATD	GMI_AD[11:8]
ATE	GMI_AD[15:12]
CSI	CSI_CLKAN CSI_CLKAP CSI_CLKBN CSI_CLKBP CSI_D1AN CSI_D1AP CSI_D2AN CSI_D2AP CSI_D1BN CSI_D1BP
CDEV1	DAP_MCLK1
CDEV2	DAP_MCLK2
CRTP	CRT_HSYNC CRT_VSYNC
CSUS	VI_MCLK
DAP1	DAP1_FS DAP1_SCLK DAP1_DIN DAP1_DOUT
DAP2	DAP2_FS DAP2_SCLK DAP2_DIN DAP2_DOUT
DAP3	DAP3_FS DAP3_DIN DAP3_DOUT DAP3_SCLK
DAP4	DAP4_FS DAP4_DIN DAP4_DOUT DAP4_SCLK
DBG	JTAG_TRST_N JTAG_TDO JTAG_TMS JTAG_TCK JTAG_TDI
DDC	DDC_SCL DDC_SDA
DSI	DSI_CLKAN DSI_CLKAP DSI_D1AN DSI_D1AP DSI_D2AN DSI_D2AP
DTA	VI_D[1:0]
DTB	VI_D[11:10]
DTC	VI_VSYNC VI_HSYNC
DTD	VI_PCLK VI_D[9:2]
DTE	VI_GP0 VI_GP[6:3]
DTF	CAM_I2C_SCL CAM_I2C_SDA
GMA	GMI_AD [23:20]
GMB	GMI_WP_N
GMC	GMI_AD[19:16]
GMD	GMI_CS[1:0]_N
GME	GMI_AD[27:24]
GPU	GPIO_PU[6:0]
GPU7	JTAG_RTCK
GPV	GPIO_PV[6:4]
HDINT	HDMI_INT_N
HDMI	HDMI_RSET HDMI_TXCN HDMI_TXCP HDMI_TXD0N HDMI_TXD0P HDMI_TXD1N HDMI_TXD1P HDMI_TXD2N HDMI_TXD2P
I2CP	PWR_I2C_SCL PWR_I2C_SDA
IRRX	UART2_RTS_N
IRTX	UART2_CTS_N
KBCA	KB_ROW[2:0]
KBCB	KB_ROW[15:7]
KBCC	KB_COL[1:0]
KBCD	KB_ROW[6:3]
KBCE	KB_COL7
KBCF	KB_COL[6:2]
LCSN	LCD_CS0_N
LD0	LCD_D0
LD1	LCD_D1
LD2	LCD_D2
LD3	LCD_D3
LD4	LCD_D4
LD5	LCD_D5
LD6	LCD_D6
LD7	LCD_D7
LD8	LCD_D8
LD9	LCD_D9
LD10	LCD_D10
LD11	LCD_D11
LD12	LCD_D12
LD13	LCD_D13
LD14	LCD_D14
LD15	LCD_D15
LD16	LCD_D16
LD17	LCD_D17
LDC	LCD_DC0

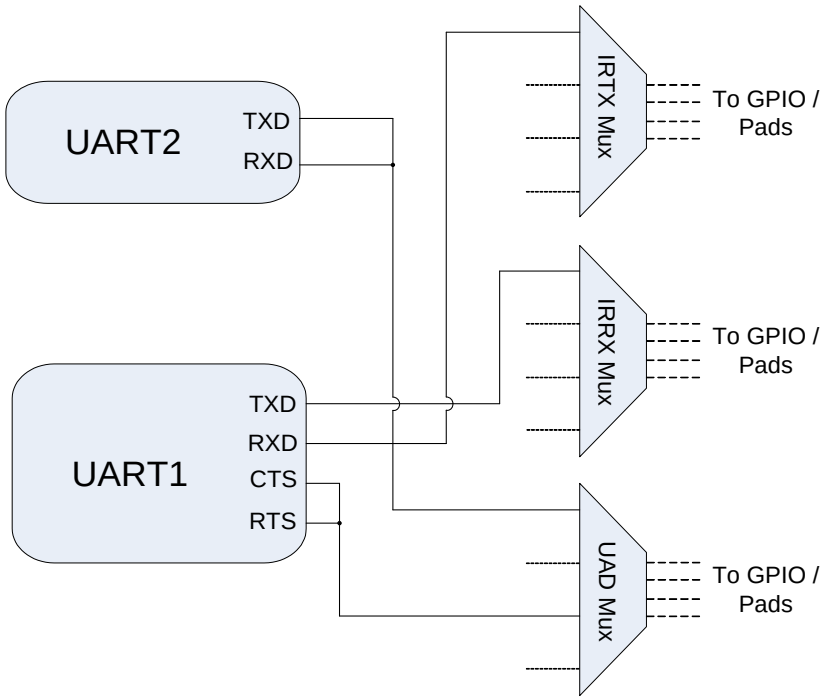
Group	Pads
LDI	LCD_D22
LHP0	LCD_D21
LHP1	LCD_D18
LHP2	LCD_D19
LHS	LCD_HSYNC
LM0	LCD_CS1_N
LM1	LCD_M1
LPP	LCD_D23
LPW0	LCD_PWR0
LPW1	LCD_PWR1
LPW2	LCD_PWR2
LSC0	LCD_PCLK
LSC1	LCD_WR_N
LSCK	LCD_SCK
LSDA	LCD_SDOUT
LSDI	LCD_SDIN
LSPI	LCD_DE
LVP0	LCD_DC1
LVP1	LCD_D20
LVS	LCD_VSYNC
MIPI	DSI_CSI_RDN DSI_CSI_RUP
OSC	XTAL_IN XTAL_OUT
OWC	OWR
PMC	CLK_32K_OUT SYS_CLK_REQ CORE_PWR_REQ CPU_PWR_REQ PWR_INT_N
PTA	GEN2_I2C_SCL GEN2_I2C_SDA
RM	GEN1_I2C_SCL GEN1_I2C_SDA
RTC	CLK_32K_IN

Group	Pads
RST	SYS_RESET_N
SDB	SDIO3_CMD
SDC	SDIO3_DAT[3:0]
SDD	SDIO3_CLK
SDIO1	SDIO1_CLK SDIO1_CMD SDIO1_DAT[3:0]
SLXA	SDIO3_DAT4
SLXC	SDIO3_DAT6
SLXD	SDIO3_DAT7
SLXK	SDIO3_DAT5
SPDI	SPDIF_IN
SPDO	SPDIF_OUT
SPIA	SPI2_MOSI
SPIB	SPI2_MISO
SPIC	SPI2_CS0_N SPI2_SCK
SPID	SPI1_MOSI
SPIE	SPI1_CS0_N SPI1_SCK
SPIF	SPI1_MISO
SPIG	SPI2_CS1_N
SPIH	SPI2_CS2_N
TST	TEST_MODE_EN
TV	VDAC_R VDAC_G VDAC_B VDAC_RSET VDAC_VREF
UAA	ULPI_DATA[3:0]
UAB	ULPI_DATA[7:4]
UAC	GPIO_PV[3:0]
UAD	UART2_RXD UART2_TXD
UCA	UART3_TXD

Group	Pads
	UART3_RXD
UCB	UART3_CTS_N UART3_RTS_N
UDA	ULPI_CLK ULPI_DIR ULPI_NXT ULPI_STP
USB	USB1_VBUS USB1_DN USB1_DP ACC1_DETECT USB3_DN USB3_DP ACC3_DETECT USB_REXT
XM2C	DDR_A[14:0] DDR_CAS_N DDR_BA[2:0] DDR_DQS0P DDR_DQS0N DDR_DQS1P DDR_DQS1N DDR_DQS2P DDR_DQS2N DDR_CKE[1:0] DDR_CLK DDR_CLK_N DDR_DM[3:0] DDR_ODT0 DDR_QUSE[3:0] DDR_RAS_N DDR_WE_N
XM2D	DDR_DQ[31:0]
XM2S	DDR_DQS3P DDR_DQS3N DDR_CS0_N DDR_CS1_N

The pad groups have been organized such that different high-level use-cases can be achieved depending on the board designer's needs. For example, the pin mux controls for the IRTX, IRRX and UAD pad groups can be configured such that the UART1 controller is available at full duplex and has a reduced set of control signals (IRTX & IRRX mux = 0, and UAD mux = 2), or such that the UART1 and UART2 controllers are available at full-duplex with software handshake only (IRTX & IRRX mux = 0, and UAD mux = 0) (see figure below).

Figure 6. UAB Configuration for Single 8-bit UART or 2 4-bit UARTs



Each controller's signals may be output to multiple pad groups, in order to maximize board design flexibility. When programming the pin mux for a specific pad group, it is necessary to ensure that no other pad groups are muxed to the same set of signals[1]; as muxing the same signal to multiple pad groups can result in system instability.

For reference, the full list of controller muxes, by pad-group, are shown below in Table 26. Additionally, several pad groups (CDEV1, CDEV2, CSUS) can be programmed to output internal Tegra 2 signals. For pads which may be configured as GPIOs, the corresponding port and pin number are provided (in <port>.<pin> notation).

1 This can happen if, for example, the power-on-reset value for a pad group refers to the same signals that have been muxed to a programmed pad group, and the power-on-reset values are not overwritten.

Figure 7. Subset of Pin Mux Possibilities for UART1 Controller's TXD, RXD, CTS and RTS Signals

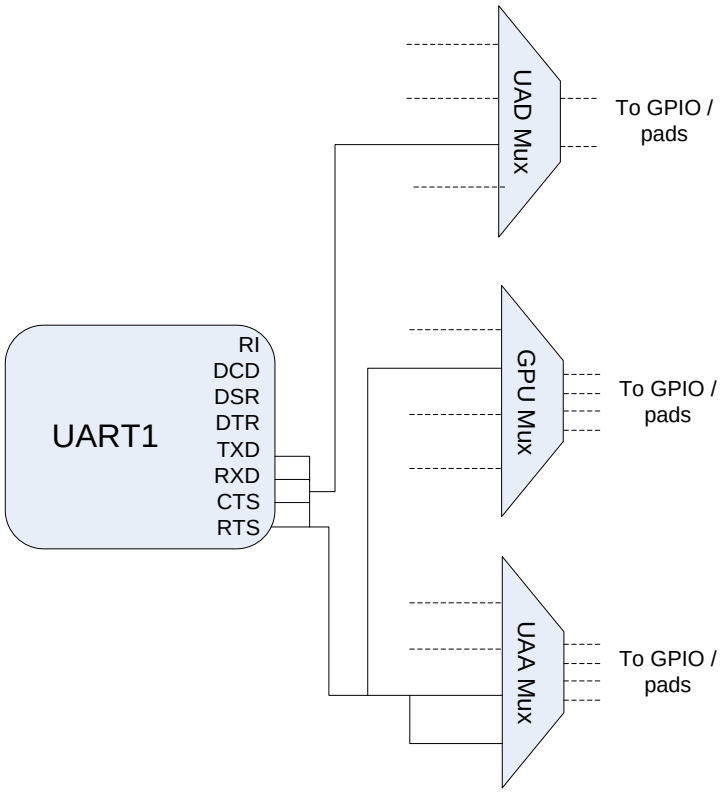


Table 26. List of all Mux Options for each Pad Group

Group	Pins	Config 0	Config 1	Config 2	Config 3	GPIO
ATA	GMI_CS6_N		NAND_CE4_N	GMI_CS6_N		I.03
	GMI_CS7_N		NAND_CE6_N	GMI_CS7_N		I.06
	GMI_RST_N	IDE_RESET	NAND_CLE	GMI_RST_N		I.04
ATB	GMI_CS5_N	IDE_DMARQ	NAND_CE5_N	GMI_CS5_N	SDIO4_CLK	I.02
	GMI_DPD	IDE_HDMACK	NAND_CLE	GMI_DPD	SDIO4_CMD	T.07
ATC	GMI_IORDY	IDE_IRQ	NAND_CE3_N	GMI_IORDY		I.05
	GMI_WAIT	IDE_IORDY	NAND_BSY0	GMI_WAIT	SDIO4_CMD	I.07
	GMI_ADV_N	IDE_A0	NAND_ALE	GMI_ADV_N		K.00
	GMI_CLK	IDE_A1	NAND_CLE	GMI_CLK	SDIO4_CLK	K.01
	GMI_CS2_N	IDE_CS0	NAND_CE0 _N	GMI_CS2_N		K.03
	GMI_CS3_N	IDE_CS1	NAND_CE1_N	GMI_CS3_N		K.04
	GMI_CS4_N	IDE_A2	NAND_CE2_N	GMI_CS4_N		K.02
	GMI_AD0	IDE_D0	NAND_D0	GMI_AD0	SDIO4_DAT1	G.00
	GMI_AD1	IDE_D1	NAND_D1	GMI_AD1	SDIO4_DAT3	G.01
	GMI_AD2	IDE_D2	NAND_D2	GMI_AD2	SDIO4_DAT5	G.02
	GMI_AD3	IDE_D3	NAND_D3	GMI_AD3	SDIO4_DAT7	G.03
	GMI_AD4	IDE_D4	NAND_D4	GMI_AD4		G.04
	GMI_AD5	IDE_D5	NAND_D5	GMI_AD5		G.05
	GMI_AD6	IDE_D6	NAND_D6	GMI_AD6		G.06
	GMI_AD7	IDE_D7	NAND_D7	GMI_AD7		G.07
	GMI_WR_N	IDE_WR_N	NAND_WE _N	GMI_WR_N		I.00
	GMI_OE_N	IDE_OE_N	NAND_RE _N	GMI_OE_N		I.01

Group	Pins	Config 0	Config 1	Config 2	Config 3	GPIO
ATD	GMI_AD08 GMI_AD09 GMI_AD10 GMI_AD11	IDE_D8 IDE_D9 IDE_D10 IDE_D11	NAND_D8 NAND_D9 NAND_D10 NAND_D11	GMI_AD08 GMI_AD09 GMI_AD10 GMI_AD11	SDIO4_DAT0 SDIO4_DAT2 SDIO4_DAT4 SDIO4_DAT6	H.00 H.01 H.02 H.03
ATE	GMI_AD12 GMI_AD13 GMI_AD14 GMI_AD15	IDE_D12 IDE_D13 IDE_D14 IDE_D15	NAND_D12 NAND_D13 NAND_D14 NAND_D15	GMI_AD12 GMI_AD13 GMI_AD14 GMI_AD15		H.04 H.05 H.06 H.07
CDEV1	DAP_MCLK1	OSC	PLLA_OUT	PLLM_OUT1	AUDIO_SYNC	W.04
CDEV2	DAP_MCLK2	OSC	AHB_CLK	APB_CLK	PLLP_OUT4	W.05
CRTP	CRT_HSYNC CRT_VSYNC	CRT_HSYNC CRT_VSYNC				
CSI	CSI_CLKAP CSI_CLKAN CSI_CLKBP CSI_CLKBN CSI_D1AP CSI_D1AN CSI_D2AP CSI_D2AN CSI_D1BP CSI_D1BN	CSI_CLKAP CSI_CLKAN CSI_CLKBP CSI_CLKBN CSI_D1AP CSI_D1AN CSI_D2AP CSI_D2AN CSI_D1BP CSI_D1BN				
CSUS	VI_MCLK	PLLC_OUT1	PLLP_OUT2	PLLP_OUT3	VI_CLK	T.01
DAP1	DAP1_FS DAP1_DIN DAP1_DOUT DAP1_SCLK	DAP1_FS DAP1_DIN DAP1_DOUT DAP1_SCLK		GMI_D28 GMI_D29 GMI_D30 GMI_D31	SDIO2_CMD SDIO2_DAT0 SDIO2_DAT1 SDIO2_SCLK	N.00 N.01 N.02 N.03
DAP2	DAP2_FS DAP2_SCLK DAP2_DIN DAP2_DOUT	DAP2_FS DAP2_SCLK DAP2_DIN DAP2_DOUT	TWC_CS_N TWC_CLK TWC_DIN TWC_DO		GMI_A17 GMI_A18 GMI_A19 GMI_A20	A.02 A.03 A.04 A.05
DAP3	DAP3_FS DAP3_DIN DAP3_DOUT DAP3_SCLK	DAP3_FS DAP3_DIN DAP3_DOUT DAP3_SCLK				P.00 P.01 P.02 P.03
DAP4	DAP4_FS DAP4_DIN DAP4_DOUT DAP4_SCLK	DAP4_FS DAP4_DIN DAP4_DOUT DAP4_SCLK		GMI_A13 GMI_A14 GMI_A15 GMI_A16		P.04 P.05 P.06 P.07
DBG	JTAG_TRST_N JTAG_TDO JTAG_TMS JTAG_TCK JTAG_TDI	JTAG_TRST_N JTAG_TDO JTAG_TMS JTAG_TCK JTAG_TDI				
DDC	DDC_SCL DDC_SDA	GEN2_I2C_SCL GEN2_I2C_SDA				
DSI	DSI_CLKAN DSI_CLKAP DSI_D1AN DSI_D1AP DSI_D2AN DSI_D2AP	DSI_CLKAN DSI_CLKAP DSI_D1AN DSI_D1AP DSI_D2AN DSI_D2AP				

Group	Pins	Config 0	Config 1	Config 2	Config 3	GPIO
DTA	VI_D0 VI_D1		 SDIO2_CMD	VI_D0 VI_D1		T.04 D.05
DTB	VI_D10 VI_D11			VI_D10 VI_D11	SPI1_MOSI SPI1_MISO	T.02 T.03
DTC	VI_HSYNC VI_VSYNC			VI_HSYNC VI_VSYNC		D.07 D.06
DTD	VI_PCLK VI_D[9:2]		SDIO2_CLK SDIO2_DAT[7:0]	VI_PCLK VI_D[9:2]		T.00 L.[07:00]
DTE	VI_GP0 VI_GP3 VI_GP4 VI_GP5 VI_GP6			VGP0 VGP3 VGP4 VGP5 VGP6	 SPI1_SCK SPI1_CS0	BB.01 BB.04 BB.05 D.02 A.00
DTF	CAM_I2C_SCL CAM_I2C_SDA	GEN3_I2C_SCL GEN3_I2C_SDA		VI_GP1 VI_GP2		BB.02 BB.03
GMA	GMI_AD20 GMI_AD21 GMI_AD22 GMI_AD23	UART5_TXD UART5_RXD UART5_CTS_N UART5_RTS_N	SPI3_SCK SPI3_MOSI SPI3_MISO SPI3_CS0_N	GMI_AD20 GMI_AD21 GMI_AD22 GMI_AD23	SDIO4_DAT0 SDIO4_DAT1 SDIO4_DAT2 SDIO4_DAT3	AA.00 AA.01 AA.02 AA.03
GMB	GMI_WP_N	IDE_IRQ_N	NAND_CE5_N	GMI_WP_N	GMI_INT1	C.07
GMC	GMI_AD16 GMI_AD17 GMI_AD18 GMI_AD19	UART4_TXD UART4_RXD UART4_CTS_N UART4_RTS_N	SPI4_SCK SPI4_MOSI SPI4_MISO SPI4_CS1_N	GMI_AD16 GMI_AD17 GMI_AD18 GMI_AD19	GMI_INT2 SFLASH_DOUT SFLASH_DIN	J.07 B.00 B.01 K.07
GMD	GMI_CS0_N GMI_CS1_N		NAND_CE6_N NAND_CE7_N	GMI_CS0_N GMI_CS1_N	SFLASH_CS0_N SFLASH_CLK	J.00 J.02
GME	GMI_AD24 GMI_AD25 GMI_AD26 GMI_AD27		DAP5_FS DAP5_DIN DAP5_DOUT DAP5_SCLK	GMI_AD24 GMI_AD25 GMI_AD26 GMI_AD27	SDIO4_DAT4 SDIO4_DAT5 SDIO4_DAT6 SDIO4_DAT7	AA.04 AA.05 AA.06 AA.07
GPU	GPIO_PU0 GPIO_PU1 GPIO_PU2 GPIO_PU3 GPIO_PU4 GPIO_PU5 GPIO_PU6	 PM3_PWM0 PM3_PWM1 PM3_PWM2 PM3_PWM3	UART1_TXD UART1_RXD UART1_CTS_N UART1_RTS_N UART1_DTR_N UART1_RI_N UART1_DSR_N	GMI_A6 GMI_A7 GMI_A8 GMI_A9 GMI_A10 GMI_A11 GMI_A12		U.00 U.01 U.02 U.03 U.04 U.05 U.06
GPU7	JTAG_RTCK	JTAG_RTCK				U.07
GPV	GPIO_PV4 GPIO_PV5 GPIO_PV6	PEX_PRSNT0_N PEX_RST0_N PEX_CLKREQ0_N				V.04 V.05 V.06
HDINT	HDMI_INT_N					N.07
HDMI	HDMI_RSET HDMI_TXD0P HDMI_TXD0N HDMI_TXD1P HDMI_TXD1N HDMI_TXD2P HDMI_TXD2N HDMI_TXCP HDMI_TXCN	HDMI_RSET HDMI_TXD0P HDMI_TXD0N HDMI_TXD1P HDMI_TXD1N HDMI_TXD2P HDMI_TXD2N HDMI_TXCP HDMI_TXCN				
I2CP	PWR_I2C_SCL PWR_I2C_SDA	PWR_I2C_SCL PWR_I2C_SDA	I2CPMU_CLK I2CPMU_DAT			Z.06 Z.07

Group	Pins	Config 0	Config 1	Config 2	Config 3	GPIO
IRRX	UART2_RTS_N	UART1_TXD	UART2_RTS_N	GMI_A0	SPI4_MISO	J.06
IRTX	UART2_CTS_N	UART1_RXD	UART2_CTS_N	GMI_A1	SPI4_CS1_N	J.05
KBCA	KB_ROW0 KB_ROW [2:1]	KB_ROW0 KB_ROW [2:1]	NAND_D0 NAND_D [2:1]	 SDIO2_DAT[5 :4]		R.00 R.[02:01]
KBCB	KB_ROW7 KB_ROW8 KB_ROW9 KB_ROW10 KB_ROW11 KB_ROW12 KB_ROW13 KB_ROW14 KB_ROW15	KB_ROW7 KB_ROW8 KB_ROW9 KB_ROW10 KB_ROW11 KB_ROW12 KB_ROW13 KB_ROW14 KB_ROW15	NAND_D7 NAND_D8 NAND_D9 NAND_D10 NAND_D11 NAND_D12 NAND_D13 NAND_D14 NAND_D15	SDIO2_DAT0 SDIO2_DAT1 SDIO2_DAT2 SDIO2_DAT3 TRACEDATA0 TRACEDATA1 TRACEDATA2		R.07 S.00 S.01 S.02 S.03 S.04 S.05 S.06 S.07
KBCC	KB_COL0 KB_COL1	KB_COL0 KB_COL1	NAND_BSY0 NAND_ALE	TRACEDATA3 TRACEDATA4		Q.00 Q.01
KBCD	KB_ROW3 KB_ROW4 KB_ROW5 KB_ROW6	KB_ROW3 KB_ROW4 KB_ROW5 KB_ROW6	NAND_D3 NAND_D4 NAND_D5 NAND_D6	SDIO2_DAT6 SDIO2_DAT7 SDIO2_SCLK SDIO2_CMD		R.03 R.04 R.05 R.06
KBCE	KB_COL7	KB_COL7	NAND_CE2	OWR_PCTLZ		Q.07
KBCF	KB_COL2 KB_COL3 KB_COL4 KB_COL5 KB_COL6	KB_COL2 KB_COL3 KB_COL4 KB_COL5 KB_COL6	NAND_CLE NAND_WE _N NAND_RE _N NAND_CE0 _N NAND_CE1	TRACEDATA5 TRACEDATA6 TRACEDATA7 TRACECLK TRACECTL		Q.02 Q.03 Q.04 Q.05 Q.06
LCSN	LCD_CS0_N	LCD1_CS0_N	LCD2_CS0_N	SPI3_CS2_N		N.04
LD[15:0]	LCD_D[7:0] LCD_D[15:8]	LCD1_D[7:0] LCD1_D[15:8]	LCD2_D[7:0] LCD2_D[15:8]			E.[07:00] F.[07:00]
LD[17:16]	LCD_D[17:16]	LCD1_D[17:16]	LCD2_D[17:16]			M.[01:00]
LDC	LCD_DC0	LCD1_DC0	LCD2_DC0			N.06
LDI	LCD_D22	LCD1_D22	LCD2_D22			M.06
LHP0	LCD_D21	LCD1_D21	LCD2_D21			M.05
LHP1	LCD_D18	LCD1_D18	LCD2_D18			M.02
LHP2	LCD_D19	LCD1_D19	LCD2_D19			M.03
LHS	LCD_HSYNC	LCD1_HSYNC	LCD2_HSYNC			J.03
LM0	LCD_CS1_N	LCD1_CS1_N	LCD2_CS1_N	SPI3_CS3_N		W.00
LM1	LCD_M1	LCD1_M1	LCD2_M1			W.01
LPP	LCD_D23	LCD1_D23	LCD2_D23			M.07
LPW0	LCD_PWR0	LCD1_PWR0	LCD2_PWR0	SPI3_MOSI		B.02
LPW1	LCD_PWR1	LCD1_PWR1	LCD2_PWR1			C.01
LPW2	LCD_PWR2	LCD1_PWR2	LCD2_PWR2	SPI3_MISO		C.06
LSC0	LCD_PCLK	LCD1_PCLK	LCD2_PCLK			B.03
LSC1	LCD_WR_N	LCD1_WR_N	LCD2_WR_N	SPI3_SCK		Z.03
LSCK	LCD_SCK	LCD1_SCK	LCD2_SCK	SPI3_SCK		Z.04
LSDA	LCD_SDOUT	LCD1_SDOUT	LCD2_SDOUT	SPI3_MOSI		N.05
LSDI	LCD_SDIN	LCD1_SDIN	LCD2_SDIN	SPI3_MISO		Z.02
LSPI	LCD_DE	LCD1_DE	LCD2_DE			J.01
LVP0	LCD_DC1	LCD1_DC1	LCD2_DC1			V.07
LVP1	LCD_D20	LCD1_D20	LCD2_D20			M.04
LVS	LCD_VSYNC	LCD1_VSYNC	LCD2_VSYNC			J.04

Group	Pins	Config 0	Config 1	Config 2	Config 3	GPIO
MIPI	DSI_CSI_RDN DSI_CSI_RUP	DSI_CSI_RDN DSI_CSI_RUP				
OSC	XTAL_IN XTAL_OUT	XTAL_IN XTAL_OUT				
OWC	OWR	OWR				
PMC	CLK_32K_OUT SYS_CLK_REQ CORE_PWR_REQ CPU_PWR_REQ PWR_INT_N	CLK_32K_OUT SYS_CLK_REQ CORE_PWR_REQ CPU_PWR_REQ PWR_INT_N				B.00 Z.05
PTA	GEN2_I2C_SCL GEN2_I2C_SDA	GEN2_I2C_SCL GEN2_I2C_SDA		GMI_CS6_N GMI_CS7_N		T.05 T.06
RM	GEN1_I2C_SCL GEN1_I2C_SDA	GEN1_I2C_SCL GEN1_I2C_SDA				C.04 C.05
RST	SYS_RESET_N	SYS_RESET_N				
RTC	CLK_32K_IN	CLK_32K_IN				
SDB	SDIO3_CMD	UART1_RXD	PWFM_PWM3	SDIO3_CMD	SPI2_SCK	A.07
SDC	SDIO3_DAT0 SDIO3_DAT1 SDIO3_DAT2 SDIO3_DAT3	PEX_CLKREQ1_N PEX_WAKE_N PMFM_PWM1 PMFM_PWM0	TWC_DO TWC_DIN TWC_CLK TWC_CS_N	SDIO3_DAT0 SDIO3_DAT1 SDIO3_DAT2 SDIO3_DAT3	SPI3_MISO SPI3_MOSI SPI3_CS0_N SPI3_CS1_N	B.07 B.06 B.05 B.04
SDD	SDIO3_CLK	UART1_TXD	PWFM_PWM2	SDIO3_CLK	SPI3_SCK	A.06
SDIO1	SDIO1_CLK SDIO1_CMD SDIO1_DAT0 SDIO1_DAT1 SDIO1_DAT2 SDIO1_DAT3	SDIO1_CLK SDIO1_CMD SDIO1_DAT0 SDIO1_DAT1 SDIO1_DAT2 SDIO1_DAT3		UART5_RTS_N UART5_CTS_N UART5_RXD UART5_TXD	UART1_DTR_N UART1_CTS_N UART1_RTS_N UART1_RI_N UART1_RXD UART1_TXD	Z.00 Z.01 Y.07 Y.06 Y.05 Y.04
SLXA	SDIO3_DAT4	PEX_RST1_N	SPI4_MISO	SDIO3_DAT4	SPI2_MISO	D.01
SLXC	SDIO3_DAT6	SPDIF_IN	SPI4_CS0_N	SDIO3_DAT6	SPI2_CS0_N	D.03
SLXD	SDIO3_DAT7	SPDIF_OUT	SPI4_MOSI	SDIO3_DAT7	SPI2_CS1_N	D.04
SLXK	SDIO3_DAT5	PEX_PRSNT1_N	SPI4_SCK	SDIO3_DAT5	SPI2_MOSI	D.00
SPDI	SPDIF_IN	SPDIF_IN		GEN1_I2C_SDA	SDIO2_DAT3	K.06
SPDO	SPDIF_OUT	SPDIF_OUT		GEN1_I2C_SCL	SDIO2_DAT2	K.05
SPIA	SPI2_MOSI	SPI1_MOSI	SPI2_MOSI	SPI3_MOSI	GMI_A21	X.00
SPIB	SPI2_MISO	SPI1_MISO	SPI2_ MISO	SPI3_ MISO	GMI_A22	X.01
SPIC	SPI2_CS0_N SPI2_SCK	SPI1_CS0_N SPI1_SCK	SPI2_CS0_N SPI2_SCK	SPI3_CS1_N SPI3_SCK	GMI_A24 GMI_A23	X.03 X.02
SPID	SPI1_MOSI	SPI2_ MOSI	SPI1_ MOSI	SPI2_ MOSI	GMI_A25	X.04
SPIE	SPI1_CS0_N SPI1_SCK	SPI2_CS1_N SPI2_SCK	SPI1_CS0_N SPI1_SCK	SPI2_CS1_N SPI2_SCK	GMI_A27 GMI_A26	X.06 X.05
SPIF	SPI1_MISO	SPI3_MISO	SPI1_MISO	SPI2_MISO		X.07
SPIG	SPI2_CS1_N	SPI3_SCK	SPI2_CS1_N	SPI2_CS2_N	GEN_I2C_SCL	W.02
SPIH	SPI2_CS2_N	SPI3_CS0_N	SPI2_CS2_N	SPI2_CS3_N	GEN_I2C_SDA	W.03
TST	TEST_MODE_EN	TEST_MODE_EN				
TV	VDAC_R VDAC_G VDAC_B VDAC_RSET VDAC_VREF	VDAC_R VDAC_G VDAC_B VDAC_RSET VDAC_VREF				

Group	Pins	Config 0	Config 1	Config 2	Config 3	GPIO
UAA	ULPI_DATA0 ULPI_DATA1 ULPI_DATA2 ULPI_DATA3	SPI3_MOSI SPI3_MISO SPI3_SCK SPI3_CS1_N	HSI_TX_DATA HSI_RX_DATA HSI_TX_READY HSI_RX_READY	UART1_TXD UART1_RXD UART1_CTS_N UART1_RTS_N	ULPI_DATA0 ULPI_DATA1 ULPI_DATA2 ULPI_DATA3	O.01 O.02 O.03 O.04
UAB	ULPI_DATA4 ULPI_DATA5 ULPI_DATA6 ULPI_DATA7	SPI2_MOSI SPI2_MISO SPI2_SCK SPI2_CS1_N	HSI_RX_WAKE HSI_TX_WAKE HSI_RX_FLAG HSI_TX_FLAG	UART1_RI_N UART1_DCD_N UART1_DSR_N UART1_DTR_N	ULPI_DATA4 ULPI_DATA5 ULPI_DATA6 ULPI_DATA7	O.05 O.06 O.07 O.00
UAC	GPIO_PV0 GPIO_PV1 GPIO_PV2 GPIO_PV3	 OWR_PCTIZ CLK12M_OUT				V.00 V.01 V.02 V.03
UAD	UART2_RXD UART2_TXD	UART2_RXD UART2_TXD	SPDIF_OUT SPDIF_IN	UART1_CTS_N UART1_RTS_N	SPI4_MOSI SPI4_SCK	C.03 C.02
UCA	UART3_RXD UART3_TXD	UART3_RXD UART3_TXD		GMI_A3 GMI_A2		W.07 W.06
UCB	UART3_CTS_N UART3_RTS_N	UART3_CTS_N UART3_RTS_N	 PWFM_PWM0	GMI_A5 GMI_A4		A.01 C.00
UDA	ULPI_CLK ULPI_DIR ULPI_NXT ULPI_STP	SPI1_MOSI SPI1_MISO SPI1_SCK SPI1_CS0_N		UART4_TXD UART4_RXD UART4_CTS_N UART4_RTS_N	ULPI_CLK ULPI_DIR ULPI_NXT ULPI_STP	Y.00 Y.01 Y.02 Y.03
USB	USB1_VBUS USB1_DN USB1_DP ACC1_DETECT USB3_DN USB3_DP ACC3_DETECT USB_REXT	USB1_VBUS USB1_DN USB1_DP ACC1_DETECT USB3_DN USB3_DP ACC3_DETECT USB_REXT				
XM2C	DDR_A[14:0] DDR_CAS_N DDR_BA[2:0] DDR_DQS0P DDR_DQS0N DDR_DQS1P DDR_DQS1N DDR_DQS2P DDR_DQS2N DDR_CKE[1:0] DDR_CLK DDR_CLK_N DDR_DM[3:0] DDR_ODT DDR_QUSE[3:0] DDR_RAS_N DDR_WE_N	DDR_A[14:0] DDR_CAS_N DDR_BA[2:0] DDR_DQS0P DDR_DQS0N DDR_DQS1P DDR_DQS1N DDR_DQS2P DDR_DQS2N DDR_CKE[1:0] DDR_CLK DDR_CLK_N DDR_DM[3:0] DDR_ODT DDR_QUSE[3:0] DDR_RAS_N DDR_WE_N				
XM2D	DDR_DQ[31:0]	DDR_DQ[31:0]				
XM2S	DDR_DQS3P DDR_DQS3N DDR_CS0_N DDR_CS1_N	DDR_DQS3P DDR_DQS3N DDR_CS0_N DDR_CS1_N				

8.2 Programming Interface

With the exception of the per-pad GPIO selection controls, all the registers for controlling pin muxing are located in the APB_MISC register aperture. Most pad groups have three controls: TRISTATE, PULLUPDOWN and PIN_MUX_CTL.

8.2.1 Tristate Control

The TRISTATE control enables software to tristate all output pads in the entire pad group when the pads are not needed for either their assigned function or as GPIOs. Each pad group's TRISTATE control can be programmed to one of two states: NORMAL or TRISTATE.

When a pad group is programmed to the NORMAL state, all of the pads in the group will be actively driven, even when the pin mux control is programmed to an invalid configuration.

When a pad group is programmed to the TRISTATE state, the entire pad group will be tristated, even if the pin mux control is programmed to an in-use controller or a pad has been configured as a GPIO.

Tristating can significantly reduce the I/O power consumption of the SoC, so the TRISTATE state should be used whenever the pads are not in-use.

8.2.2 Pullupdown Control

The PULLUPDOWN control is used to configure internal pull-up / pull-down for pad inputs, eliminating the need for external components to do this. Each pad group's PULLUPDOWN control may be programmed to one of three states: PULL_UP, PULL_DOWN, or NORMAL.

In the PULL_UP state, all inputs in the pad group will be pull-up, and in the PULL_DOWN state the pads will be pull-down.

In NORMAL state the pad will not be driven. If external pull-up / pull-down components are used for a pad group, the PULLUPDOWN control for that pad group should be programmed to either NORMAL or programmed to operate in the same direction (i.e., PULL_UP when external pull-up components are used, PULL_DOWN when external pull-down components are used). Mismatched internal and external controls will result in a significant increase in power consumption.

8.2.3 Pin_Mux_CTL Control

The PIN_MUX_CTL control is used to select the signals that should be output by each pad group (i.e., the pin mux configuration) from the list in Table 26.

As mentioned in the Pad Groups Section, it is necessary to ensure that no two pads are configured to output the same signal, as doing so can result in system instability.

Note that it is *not* sufficient to program a single conflicting pad to a GPIO state in order to avoid the conflict, nor is it sufficient to program a conflicting pad group to tristate. The only way to completely eliminate instabilities caused by conflicting pad groups is to ensure that none of the pad groups have been programmed to conflicting configurations.

8.3 Pin Mux Use Case Configuration Examples

Often, it is convenient to think of pin muxing "in reverse" – i.e., given a specific controller, which pad groups can it drive in meaningful ways. In many cases, it may be necessary to program the PIN_MUX_CTL control for multiple pad groups simultaneously in order for all of a controller's signals to be routed to a peripheral on the board.

Using the UART1 controller as an example, the following muxed configurations can be constructed:

- If an eight-wire UART is required, software can program UAA's PIN_MUX_CTL=2 and UAB's PIN_MUX_CTL=2. Alternatively, if the carrier detect signal is not used, the GPU pin group can be used, with its PIN_MUX_CTL=1.
- If a four-wire UART is required (full-duplex data plus hardware hand-shaking), software can program, IRRX's and IRTX's PIN_MUX_CTL=0, and UAD's PIN_MUX_CTL=2.

- If only a two-wire UART is required (full-duplex data with software hand-shaking), software can program, IRRX's and IRTX's PIN_MUX_CTL=0. Alternately, SDD PIN_MUX_CTL=0

Table 27 is a depiction of these 6 pin-outs of the UART1 controller, and the pin mux programming required for each.

Table 27. Supported Pin-outs for UART1 Controller

	8-wire configuration			**Alternate 8-wire configuration**		
Signal	**Pad Name**	**Group**	**Mux**	**Pad Name**	**Group**	**Mux**
UART1_TXD UART1_RXD UART1_CTS UART1_RTS UART1_RI UART1_DCD UART1_DSR UART1_DTR	UART1_TXD UART1_RXD UART1_CTS UART1_RTS	UAA	2	GPIO_PU0 GPIO_PU1 GPIO_PU2 GPIO_PU3 GPIO_PU5 GPIO_PU6 GPIO_PU4	GPU	1
	UART1_RI UART1_DCD UART1_DSR UART1_DTR	UAB	2			
	4-wire configuration			**Alternate 4-wire configuration**		
UART1_TXD UART1_RXD UART1_CTS UART1_RTS	**UART2_RTS**	**IRRX**	**0**	**None**		
	UART2_CTS	**IRTX**	**0**			
	UART2_RXD **UART2_TXD**	**UAD**	**2**			
	2-wire configuration			**Alternate 2-wire configuration**		
UART1_TXD UART1_RXD	UART2_RTS	IRRX	0	SDIO3_CLK SDIO3_CMD	SDD	0
	UART2_CTS	IRTX	0			

The next table is an example of various pin-out configurations for the SPI-3 controller, and the required pin mux programming for each.

Table 28. Supported Pin-outs for SPI3 Controller

	Primary SPI Configuration			**Primary LCD configuration**		
Signal	**Pad Name**	**Group**	**Mux**	**Pad Name**	**Group**	**Mux**
SPI3_CS0 SPI3_SCK SPI3_DIN SPI3_DOUT SPI3_CS1 SPI3_CS2 SPI3_CS3	SPI3_CS0 SPI3_SCK SPI3_DIN SPI3_DOUT	XM2A	1	LCD_WR	LSC1	2
				LCD_PWR2	LPW2	2
				LCD_PWR0	LPW0	2
				LCD_CS1	LM0	2
	Alternate LCD configuration			**UART configuration**		
SPI3_CS0 SPI3_SCK SPI3_DIN SPI3_DOUT SPI3_CS1 SPI3_CS2 SPI3_CS3	LCD_SCK	LSCK	2	UART1_CTS UART1_RXD UART1_TXD UART1_RTS	UAA	0
	LCD_SDIN	LSDI	2			
	LCD_SDOUT	LSDO	2			
	LCD_CS0	LCSN	2			
	Alternate SPI configuration			**SDIO configuration**		
SPI3_CS0 SPI3_SCK	SPI2_SCK	SPIC	2	SDIO3_DAT2	SDC	3
				SDIO3_CLK	SDD	3

	Primary SPI Configuration			Primary LCD configuration		
Signal	**Pad Name**	**Group**	**Mux**	**Pad Name**	**Group**	**Mux**
SPI3_DIN SPI3_DOUT SPI3_CS1 SPI3_CS2 SPI3_CS3	SPI2_DIN	SPIB	2	SDIO3_DAT0 SDIO3_DAT1 SDIO3_DAT3	SDC	3
	SPI2_DOUT	SPIA	2			
	SPI2_CS0	SPIC	2			

8.4 Dynamic Pin Muxing

In some cases, board layouts may be designed where multiple pin-out configurations will be muxed from a single controller, and where the device requirements ensure that only one of the configurations will be used at any given time.

An example for this would be a board design with an audio codec connected to the SPI3 controller's Primary SPI configuration, and an LCD controller connected to the SPI3 controller's Alternate LCD configuration (from Table 28 above), where the SPI controller is only used to perform peripheral configuration at initialization and shut-down times.

In order to communicate with both sets of peripherals, it is necessary to reprogram the associated PIN_MUX_CTL registers at run-time to select between the different required controller configurations. This is referred to as *dynamic pin muxing*. Use of dynamic pin muxing can create situations where it is extremely difficult to ensure that all pad groups have been programmed to non-conflicting muxes, so it should be used sparingly.

For the SPI3 controller example, for the software to send SPI commands to the audio codec, XM2A's PIN_MUX_CTL should be programmed to 1, and all of the LCD pad-groups (LSCK, LSDI, LSDA and LCSN) should be programmed to non-conflicting muxes, such as the second display controller (LCD2_*) mux (PIN_MUX_CTL=1) and tri-stated. To send commands to the LCD controller, all of the SPI3 LCD pad groups should have their PIN_MUX_CTL=2, and XM2A should be programmed to a non-conflicting mux (e.g., 3), and tri-stated.

9.0 POWER

9.1 Power Management Controller

The Power Management Controller (PMC) block interacts with an external Power Manager Unit (PMU). The PMC mostly controls the wake-up of Tegra® 2 Series devices from different sleep modes.

Sleep and deep sleep require specific logic to maintain some state and control the power domains, including signaling to the external PMU, to provide power to the main logic in the Tegra 2 Series devices. All this logic is centralized in the PMC block.

Glossary

- Frozen boot: The RTC partition power transitions from OFF to ON
- Cold boot: The main partition power transitions from OFF to ON with no previous state available, SW must construct all state from scratch.
- Warm boot: The main partition power transitions from OFF to ON with previous state available, SW checks for the preserved state integrity and restores the saved state. This is also called Deep Sleep wake-up.

9.1.1 External Interface

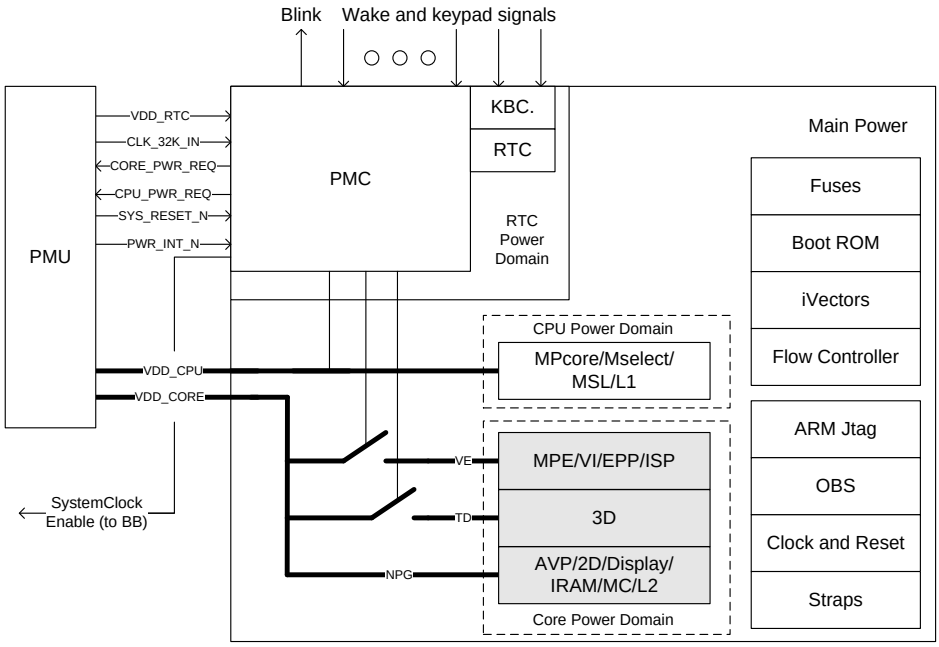
Table 29 PMC External Interface Signals

Signal Name	Type	Signal Description
SYS_RESET_N	In	Hardware reset, active low outside the chip
CORE_PWR_REQ	Out	Used to signal the external PMU device when to toggle the Main power state
CPU_PWR_REQ	Out	
Wake-up Inputs	In	Up to 29 pins designated as wake-up are used for triggering wake-up from Deep Sleep mode (23 pins on AP20 and 29 pins on Tegra 250)
SYS_CLK_REQ	Out	For systems where the system clock reference can be disabled during sleep
CLK_32K_OUT	Out	32 kHz clock out (active even during Deep Sleep). Can be used for systems that want a blinking LED during Deep Sleep

9.1.2 Functionality

PMC is a part of the RTC power domain. PMC manages the interface with the external PMU, including the hardware reset pin, the 32.768 kHz clock and the power request signal. A schematic view of the power domains inside the Tegra 2 Series devices is shown below.

Figure 8. Power Domains in Tegra 2 Series Devices



The most important aspect of PMC is wake-up and reset logic. The following diagrams show the expected behavior of PMC under three scenarios:

- Frozen boot, that is, the VDD_RTC transitioning from OFF to ON, with or without VDD_CORE and VDD_CPU also transitioning from OFF to ON
- Deep Sleep wake-up, where the PMC is responsible to track wake-up events and to request a transition of VDD_CORE and optionally VDD_CPU from OFF to ON
- Suspend mode wake-up, where the PMC is responsible for establishing the power back to the CPU partition

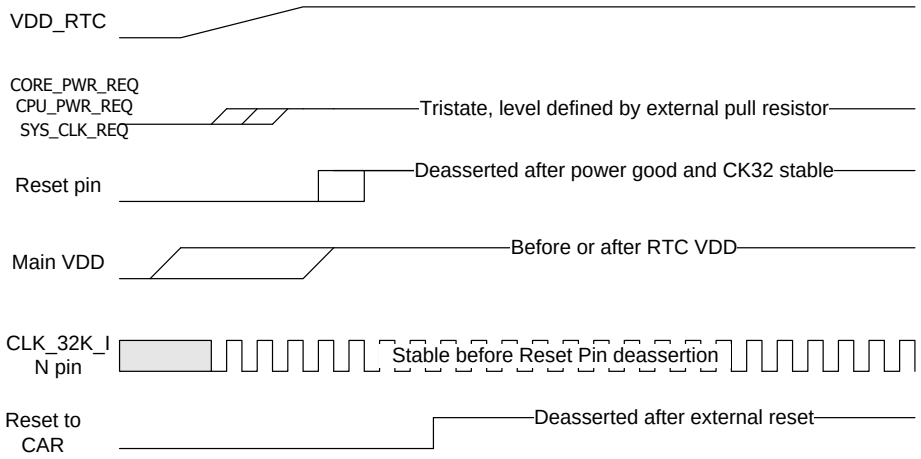
When reset is de-asserted to the Tegra 2 Series Devices, the PMC logic performs minimal processing. It forwards a reset signal and the 32.768 kHz clock to the Clock and Reset (CAR) block.

9.1.2.1 Frozen Boot Sequence

The Frozen Boot wake-up sequence happens when power has been de-asserted to the RTC domain:

1. PMU detects a turn-on condition and enables power to the VDD_RTC
2. Simultaneously to VDD_RTC enable or shortly afterwards, PMU enables power to VDD_CORE and optionally to VDD_CPU.
3. >1ms later PMU enables power to 1.8V IO rails that need to be enabled for boot
4. >1ms later PMU enables power to 2.8V/3.3V IO rails that need to be enabled for boot
5. PMU drives 32.768 kHz clock to Tegra 2 Series devices
6. >1ms later PMU de-asserts reset to Tegra 2 Series devices, 32.768 kHz clock and all power rails must be stable and at their nominal value before reset de-assertion
7. PMC block resets and de-asserts reset to CAR block
8. Tegra 2 Series devices begins boot from internal ROM

Figure 9. Frozen Boot Sequence



9.1.2.2 The Deep Sleep Wake-up Sequence

The Deep Sleep wake-up sequence is the most complex for PMC. A large part of the complexity comes from the need to operate specific part of the logic and of the IO using the VDD_RTC voltage and to avoid race conditions when going to sleep or waking up.

The corresponding IO rail for a given wakeup pin needs to be powered on in order for the wakeup to function. The VDD_CORE and VDD_CPU power rails are turned off during DEEP SLEEP. Wakeup signals are routed from the pads directly to RTC for wakeup sequence to be initiated.

A full Deep Sleep / Wake-Up sequence proceeds according to the following steps:

Setting up Deep Sleep

1. SW disables unneeded logic blocks and peripherals not needed and disables their clocks.
2. SW programs PMC with the wake-up condition
3. SW programs scratch registers with information needed to restore state after wake-up
4. SW interacts with PMC to establish the idle conditions on the IO
5. SW programs external SDRAM to enter self-refresh mode
6. SW programs the PMC to enter Deep Sleep
7. PMC asserts the clamping signals for all power gated islands
8. PMC de-asserts power request (polarity previously programmed)
9. PMU removes VDD_CORE and VDD_CPU, Deep Sleep entered

Note: To Power-off VDD_CPU:

1. Assert resets to CPU via CLK_RST_CONTROLLER_RST_CPU_CMPLX_SET
2. Disable clock to CPU via CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0_CLK_ENB_CPU
3. Remove VDD_CPU

Deep Sleep Wakeup

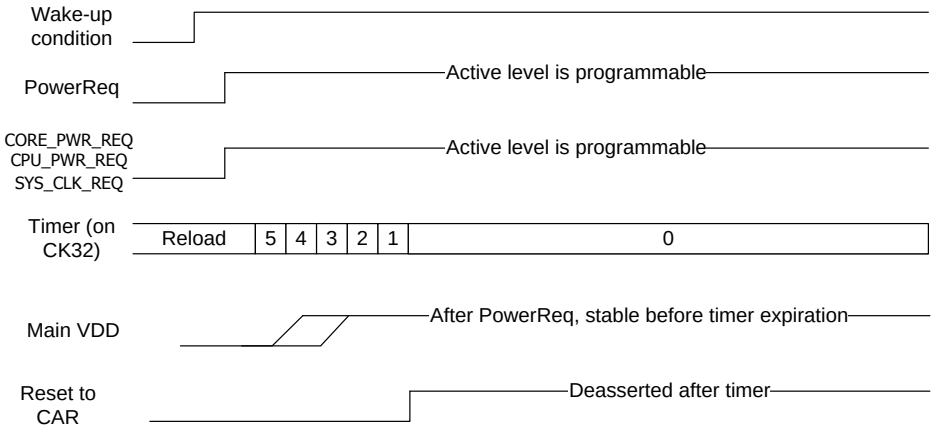
1. PMC detects a wake-up condition and latches the condition into a register
2. PMC asserts the power request signal, asserts reset to CAR and starts timer (power good delay)
3. PMU restores VDD_CORE and optionally VDD_CPU

4. Timer expires, PMC releases the reset to main

Note: To Power-on VDD_CPU:

1. Restore VDD_CPU
2. Enable clock to CPU via CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0_CLK_ENB_CPU
3. If using PLLX as CPU clock source is desired, configure/enable PLLX and switch to PLLX (this step can be done after step 4).
4. Deassert resets to CPU via CLK_RST_CONTROLLER_RST_CPU_CMPLX_CLR

Figure 10. Deep Sleep Wake-Up Sequence

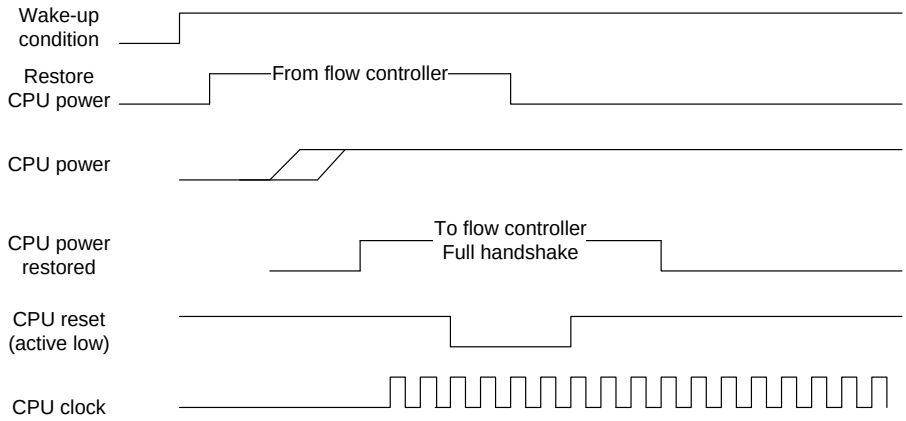


9.1.2.3 Suspend Mode Wake-up Sequence

The Suspend Mode wake-up sequence is simpler. PMC needs to wake-up CPU by removing the power gating:

1. SW configures flow controller with the Suspend Mode wake-up conditions.
2. Flow controller interacts with CPU to ensure that no transactions are in flight
3. Flow controller indicates to CAR to reset CPU
4. PMC clamps CPU signals and removes power from CPU
5. Flow controller detects Suspend mode wake-up condition
6. Flow controller sends a trigger event to PMC
7. PMC restores power to CPU
8. PMC sends an acknowledge to flow controller
9. Flow controller restores clock to CPU
10. CAR asserts reset to CPU
11. PMC removes clamping to the CPU
12. CAR de-asserts reset to CPU

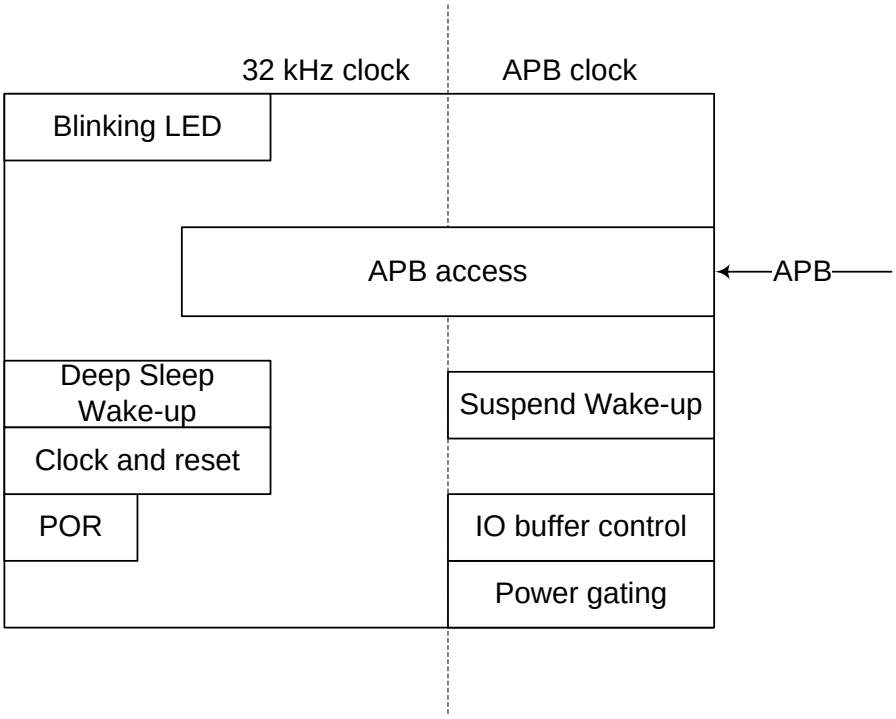
Figure 11. Suspend Mode Wake-Up Sequence



9.1.3 Reference Block Decomposition

The next figure shows a reference decomposition of PMC to simplify the description. The design uses a different functional partitioning.

Figure 12. Reference Block Decomposition Diagram



9.1.3.1 PMC

The blocks inside PMC are:

- Wake-up logic for Deep Sleep and Suspend modes
- Power gating logic
- External clock and reset
- Blinking LED

- IO buffer control
- APB interface

9.1.4 Deep Sleep Entry Logic

There are two ways to enter Deep Sleep mode:

- Writing into the DPD register
- Interaction with the power gating logic when the DPD trigger enable (side effect) bit is set.

The second method is the recommended usage as described before. The exact sequence has been described in more detail earlier in this section.

Note that entering Deep sleep mode should happen after an ordered shutdown of Tegra 2 Series devices. As part of the shutdown procedure, external interfaces must be placed in their idle condition. PMC provides the logic that ensures that interfaces maintain their idle state during Deep Sleep itself. The SW is responsible to sample the idle state by writing to the DPD sample bit before entering Deep Sleep mode.

SW must also ensure that IO paths that must remain active during Deep Sleep mode are identified as such, by correct configuration of ownership bits in a PMC register. These ownership bits are present for the signals used by KBC (where the size of the scan matrix is defined by the customer, the exact required size is reflected in the ownership bits) and for the clock request signal (that may not be required depending on the customer clock logic).

SW must also ensure that wake-up conditions are programmed properly. Wake-up level and wake-up mask should reflect expected wake-up events. Power Good Timer and Power Down Timer must be set according to Power Management Unit Specification.

The Deep Sleep mode entry/wake-up logic operates largely on the 32.768 kHz clock, so SW must respect a minimal delay between a Deep Sleep mode exit and before issuing the next Deep Sleep mode entry command. This delay is of the order of 2 periods of the 32.768 kHz clock. This should never be a problem given that the code to restore the processor state requires much more time.

9.1.5 Deep Sleep Wake-Up Logic

The wake-up logic performs the following tasks:

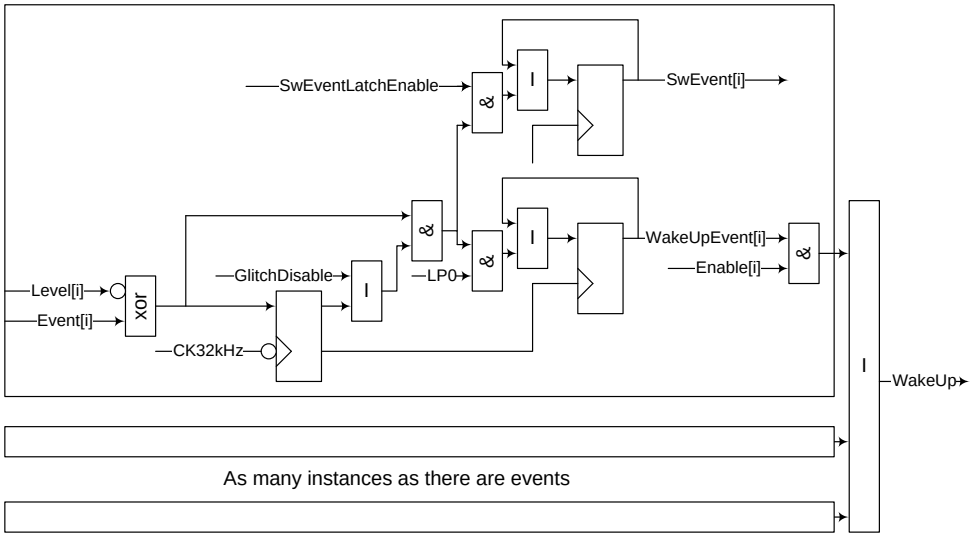
- Tracks the state of a programmable set of wake-up conditions
- Detects a wake-up condition
- Starts the sequence of action required by the Deep Sleep mode wake-up

A wake-up event is either a change of level on one in a set of specified pins or an assertion of RTC or KBC interrupts.

The logic that tracks for change of levels in external pins contains a glitch filter that may be optionally disabled. Internal interrupts don't require deglitching logic, but are passed through the same logic, as this is also the synchronization logic. A conceptual schematic is shown in the next figure. The different stages are:

- Normalizing the active level to high by inverting the wake-up signals identified as active low
- Latching either the direct signal or a deglitched version of it (global enable)
- Masking the latched signals to generate the wake-up signal

Figure 13. Deep Sleep Wake-up Logical Circuit



Note that the WakeUp signal shown is not directly the CORE_PWR_REQ signal, and there are two further processing steps defining the CORE_PWR_REQ signal:

- The polarity of the CORE_PWR_REQ signal is programmable
- CORE_PWR_REQ signal is disabled at reset. It should be enabled only after polarity is set accordingly to PMU specification
- Furthermore, there is a System Clock enable signal that is equal to CORE_PWR_REQ, but with its own independent control bits for polarity and output enable.

The set of latched wake-up events are readable by SW once the processors are active. The logic accumulates all events that take place while in Deep Sleep mode. Normally only one enabled event is present in the set, but simultaneous events could take place. Each bit of the latched wake-up event register is cleared by writing a 1b to it.

A second set of latches is also present, with the same structure as the wake-up event register but under software control. The most important difference between the two sets is that events that occur outside of the Deep Sleep mode will be captured in the second set, including:

- Events that could occur during the Deep Sleep mode transition itself, i.e. before the HW enters the Deep Sleep state, but after SW has instructed HW to enter Deep Sleep.
- Events that occur after the HW wake-up, but before SW has enabled the peripheral functions in the NPG that process the signals tied to the wake-up events, i.e. the GPIO event logic.

9.1.5.1 Deep Sleep Wake-up Event Input Path

The wake-up logic as defined, recognizes wake-up signals matching the programmed wake-up active level that exceed certain time limits and rejects events less than another limit as defined in the following table.

Table 30. Deep Sleep Wake-up Logic

Glitch disable bit	Never wake-up if event is less than	Always wake-up if more than
0 (Glitch reject)	0.5 period of 32.768 kHz clock = 15.25 us	1.5 period of 32.768 kHz clock = 45.755 us
1 (No glitch reject)	0 period of 32.768 kHz clock = 0 us	1 period of 32.768 kHz clock = 30.50 us

Between these two values, the probability of detection increases linearly from 0 to 1 when events are asynchronous to the 32.768 kHz clock. The latency delay is also a probabilistic function, varying linearly between the two limits presented in the above table.

Note that the logic doesn't contain any formal synchronization stage and so theoretically metastability cannot be precluded at this time. However the MTBF linked to metastability will be insignificant given the very low active duty of the wake-up circuitry and the low frequency of the sample clock (32.768 kHz). Adding an explicit synchronization stage increases the latency of the circuit.

9.1.6 Suspend Wake-up Logic

The only Suspend mode wake-up specific logic in PMC is related to the power gating logic as described previously.

9.1.7 Power Gating Logic

The power gating logic is a simple programmable sequencer. When the power gating logic is triggered, it sequences the set of power gating signals with a programmable period (measured in cycles) between the toggling of consecutive power gating signals.

There are three power gated domains. For all of them the trigger can be a register write (see register definition). The CPU power gated domain has an extra trigger coming from the flow controller. PMC and flow controller provides a full handshake, with PMC providing an acknowledge back to flow controller when the CPU power has completed its transition (this would normally only be for OFF to ON transition at Suspend mode wake-up).

The power gating logic operates on APB clock. The timing sequence may be dependent on the exact frequency, with a hardware limit that the minimum transition period is at least 64 ns.

The power gating logic may be triggered either by direct register writes using the toggle register or by interaction with the flow controller. The second control path is used when entering Deep Sleep or Suspend mode to allow for an orderly shutdown of the processor. The sequence can be summarized in the following manner:

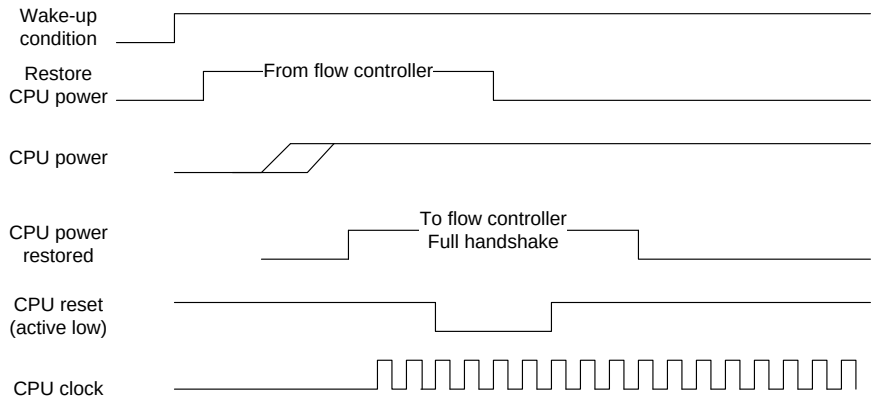
1. The CPU decides to enter either Deep Sleep or Suspend mode.
2. In both cases, the power to the CPU must be gated off
3. CPU does any required clean up job. This is especially important for Deep Sleep (all interfaces in idle, save context, etc.)
4. CPU informs flow controller it wants to remove its power, with two variants:
5. For Suspend mode, CPU also defines the set of events that will wake up the CPU (the flow controller is not power gated together with the CPU)
6. For Deep Sleep, write to sample register preserves IO state to be driven when the chip enters Deep Sleep mode
7. For Deep Sleep, CPU must define in PMC the set of wake-up events for the whole Tegra 2 Series devices, and set the DPD trigger enable bit
8. CPU enters an endless loop
9. Flow controller asserts reset to CPU
10. Flow controller informs PMC that CPU power gating OFF is requested
11. PMC power gates the CPU OFF then informs flow controller
12. If the DPD trigger enable was set, PMC also performs an Deep Sleep entry, this will result after a while in the core power disappearing

The Deep Sleep wake-up was described before and is essentially a full reset of the core logic. The Suspend mode wake-up logic includes an interlock with the flow controller that works like this:

1. Flow controller detects the Suspend mode wake-up condition

2. Flow controller informs PMC that power gating ON is requested
3. PMC power gates the CPU ON then informs the flow controller
4. Flow controller de-asserts the reset to the CPU
5. CPU checks different status registers to establish this was in fact a Suspend mode wake-up.

Figure 14. Power Gating Logic



9.1.8 Clamping

This is related to power gating, when a partition is OFF. All signals leaving that partition are clamped to their idle value, normally 0, as most signals are active high.

Applying the clamping presents no specific problem, as all blocks into the partition being powered OFF are assumed to be in reset prior to the power gate transition.

Removing the clamping is more complex because Tegra 2 Series devices use a synchronous reset strategy, so the correct sequence is the following:

- Restore power, reset and clamping both active at that time
- Restore clocks, so that reset conditions propagate
- Remove clamping
- Remove reset

For maximum flexibility, removing the clamping can be controlled by SW. Writing to a trigger register removes the clamping for the identified set of partitions. This doesn't work for Suspend mode exit, where the CPU must be restarted by HW only.

In Suspend mode exit, there is an interlock between the CAR (Clock and Reset block) and PMC, so that the correct sequence is performed. CAR indicates to PMC when the CPU clamping may be removed. This signal is asserted after the clock to CPU is restarted. The reset to CPU will only be removed after a known delay following the request to remove the clamping. PMC removes the CPU clamping as soon as it receives the request to do so. This is similar to the interlock between PMC and flow controller, except that this is not a full handshake.

9.1.9 Scratch Registers

PMC provides SW with scratch registers that maintain information during Deep Sleep. SW uses these registers for information that must be preserved to allow a correct restore of the state context saved in SDRAM after a Deep Sleep wake-up.

There are two sets of registers, secure and normal. The normal registers are simple read-write registers, while the secure registers have added HW read and write protection bits that work like this:

- At POR reset or at main reset (including SW controlled main reset), the protection bits are reset to 0b, i.e. access is allowed, the corresponding code (in IROM) is assumed to be secure.
- The protection bits are disable bits that can only be set by SW, it is assumed that secure SW in IROM will set these bits before starting any application code. This is similar to other HW protection bits elsewhere in the Tegra 2 Series devices.
- The write protection bit, when set, disables writing into the secure scratch registers. A write transaction will not be flagged as an error, but the value of the secure scratch registers will not be altered.
- The read protection bit, when set, disables reading the value of the secure scratch registers. A read transaction will not be flagged as an error, but the return value is the all zeroes pattern.

9.1.10 Blinking

Tegra 2 Series devices support a blinking LED whenever the cell phone is not powered OFF, including in Deep Sleep mode. The blinking LED is controlled from the RTC partition.

PMC contains a counter with two programmable values indicating the On and Off periods (in multiple of 16 cycles of the 32.768 kHz clock). The counter is 16 bits long, for a maximum On or Off period of 2^{16} * 16 / 32.768 10^{3} or about 16 seconds.

9.1.11 IO Buffer Control

The logic for Deep Power Down IO buffer management ensures that the IO are maintained in an idle state while in Deep Sleep mode. The procedure has been described before and requires SW sequencing:

1. SW makes sure that the affected IO ports are in their idle state, i.e. the state that will persist during Deep Sleep
2. When SW sets the DPD sample register , the idle state of the IO control signals is captured, i.e. the driving direction and the data driven if the direction indicates output.
3. SW writes the DPD enable bit and muxes inside the IO buffers now drive the stored idle state, while the corresponding control signals coming from the core are ignored.
4. SW triggers the Deep Sleep mode, killing itself
5. HW detects the Deep Sleep wake-up condition
6. SW wakes-up and restores the state
7. SW clears the DPD sample bit
8. SW clears the DPD enable bit
9. IO control can be sequenced back to the controlling block once SW has initialized the core block to drive outputs to the same idle state they maintained before entering DPD.

PMC also contains a SW programmable register, with one bit per IO rail. Each bit disables internal logic inside the IO buffer that would otherwise consume power when the IO power is OFF. Each bit should be set by SW before shutting off the corresponding IO power. Reversely, each bit should be reset after transitioning back the IO voltage to On (and before enabling the set of interfaces using the corresponding IO rail).

PMC also contains two registers related to 3.3V IO support. Specific power detector cells indicate per IO voltage domain if their supply voltage is low or high. This signal must be reflected in a corresponding signal of the IO pads part of that IO voltage domain. The detector cells consume significant power, so they cannot be left on all the time. PMC contains a register enabling the detector cells. The output of the detector cell can then be sampled and written in another register linked to the control signal going to each IO voltage domain.

9.1.12 Pin Muxing

Some of the signals controlled by PMC may not be required in certain customer designs, especially:

- The system clock may not support an enable function, so the system clock enable (SYS_CLK_REQ) is not used

- The system may not require a LED active in Deep Sleep or another device (BB) may drive the LED, so the power LED is not used.

It is important to allow the corresponding balls to be reused for other purposes via pin muxing. This is not directly a responsibility of PMC but the correct pin muxing must be present. PMC provides a set of ownership registers that control if the RTC logic (PMC or KBC) has control of pins with multiple uses (KBC, PMC or typically GPIO), some of which are not controlled by RTC logic. The ownership bit controls a final stage of muxing on top of the more general pin muxing stage.

As an important special case, the system clock enable is one of the Deep Sleep wake-up event pin. Obviously usage as a wake-up event pin is incompatible with usage as a system clock enable pin.

9.1.13 APB Access

The APB access logic is complicated by the need to not unduly block the bus for extended periods of time even when accessing registers that are logically in the 32 kHz domain. The following principles are used to avoid any delay when accessing registers logically in the 32 kHz domain:

- Write operations always take place in the APB clock domain, i.e. the corresponding registers are placed in the APB clock domain and passed as wires to the 32 kHz domain where they are used without further synchronization. This only works because values passed in this manner are considered essentially static. There is an insignificant probability that the 32 kHz logic sees an intermediate value of a multiple bit field, but an enumeration of the affected fields shows that this can't result in any significant problem at the system level.
- Read operations are to shadow registers in the APB clock domain. The shadow registers continuously reflect the values of the equivalent registers found in the 32 kHz clock domain. Only registers reflecting status present in the 32 kHz domain are affected.

9.1.14 Register Definition for PMC

To speed up operation, this register file operates in the local peripheral interface bus domain (APB) rather than in the 32 KHz clock domain used for PMC processing.

Registers in PMC control entering/exiting Deep Sleep /Suspend mode, power detect, and IO power functions. They also control CORE_PWR_REQ, SYS_CLK_REQ, and LED_BLINK pins.

The following registers should be programmed/read for different power events in order for PMC to function properly.

IMPORTANT:	**NO_IO_POWER REGISTER (APBDEV_PMC_NO_IOPOWER_0)** This register is used to disable pads which have IO power turned off. When an IO 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 IO power rail is stable and you are ready to enable some interface on the IO power rail. Leaving no_io_power set to 0 **for greater than 500ms** when rail is off may reduce the lifetime of the pad. Setting it to 1 when power rail is on will disable pad from voltage stress.

BEFORE/ON ENTERING DEEP SLEEP MODE

- PMC Control Register (bits PWRREQ_POLARITY, PWRREQ_OE, SYSCLK_POLARITY,SYSCLK_OE, LATCHWAKE_EN)
- WAKE_MASK
- WAKE_LVL
- DPD_PADS_ORIDE (for required overrides)
- SCRATCH23
- PWRGOOD_TIMER
- DPD_SAMPLE

- DPD_ENABLE

AFTER WAKE-UP FROM DEEP SLEEP

- WAKE_STATUS
- SW_WAKE_SATUS
- DPD_ENABLE
- PMC Control Register (bit LATCHWAKE_EN)

BEFORE/ON POWERING DOWN PARTITIONS

- PWRGATE_STATUS
- PWRGATE_TIMER_OFF (for power down)
- PWRGATE_TIMER_ON (for power up)
- 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.

9.1.15 PMC Registers

9.1.15.1 APBDEV_PMC_CNTRL_0

Base offset in APB device space.

This register controls clock to KBC, RTC, reset to car, KBC, RTC. It also manages polarity and output enable of sys_clk_req pin and core_pwr_req pin at reset both are disabled since PMU properties are unknown.

Auxiliary functions - enables blinking functions, and software wake-up latching.

PMC Control Register

Offset: 000h | Read/Write: R/W | Reset: 0b0000000000000000

Bit	Reset	Description
18	0x0	FUSE_OVERRIDE: Fuse override 0 = DISABLE 1 = ENABLE
17	0x0	INTR_POLARITY: Inverts INTR polarity 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
16	0x0	CPUPWRREQ_OE: Power request output enable. resets to tri-state 0 = DISABLE 1 = ENABLE
15	0x0	CPUPWRREQ_POLARITY: Inverts power request polarity 0 = NORMAL 1 = INVERT
14	0x0	SIDE_EFFECT_LP0: when set causes side effect of entering Deep Sleep after powering down CPU 0 = DISABLE 1 = ENABLE
13	0x0	AOINIT: AO(Always On or RTC domain) initialized purely SW diagnostic and interpretation 0 = NOTDONE 1 = DONE
12	0x0	PWRGATE_DIS: Disable power gating 0 = DISABLE 1 = ENABLE
11	0x0	SYSCLK_OE: Enables output of system enable clock 0 = DISABLE 1 = ENABLE
10	0x0	SYSCLK_POLARITY: Inverts SYS_CLK_REQ enable polarity 0 = NORMAL 1 = INVERT
9	0x0	PWRREQ_OE: CORE_PWR_REQ output enable. resets to tristate 0 = DISABLE 1 = ENABLE
8	0x0	PWRREQ_POLARITY: Inverts CORE_PWR_REQ polarity 0 = NORMAL 1 = INVERT
7	0x0	BLINK_EN: Enables blinking counter and blink output (on CLK_32K_OUT pin) 0 = DISABLE 1 = ENABLE
6	0x0	GLITCHDET_DIS: Disable detecting glitch on wake-up event 0 = DISABLE 1 = ENABLE
5	0x0	LATCHWAKE_EN: Enables latching wakeup events on transition from 1 to 0(sequence - set to 1,set to 0) 0 = DISABLE 1 = ENABLE
4	0x0	MAIN_RST: Reset to CAR 0 = DISABLE 1 = ENABLE
3	0x0	KBC_RST: Software reset to KBC 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	0x0	RTC_RST: Software reset to RTC 0 = DISABLE 1 = ENABLE
1	0x0	RTC_CLK_DIS: Disable 32KHz clock to RTC 0 = DISABLE 1 = ENABLE
0	0x0	KBC_CLK_DIS: Disable 32KHz clock to KBC 0 = DISABLE 1 = ENABLE

9.1.15.2 APBDEV_PMC_SEC_DISABLE_0

On separate reset (same as car), disables access (read/write to secure scratch registers). Once write is disabled, secure registers cannot be written until power on reset or reset on exiting Deep Sleep.

Once read is disabled, read from scratch registers will return 0 until power on reset or reset on exiting Deep Sleep mode.

Secure Register Disable

Offset: 004h | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
3	0x0	BREAD: disable read from bondout secure registers 0 = OFF 1 = ON
2	0x0	BWRITE: disable write to bondout secure registers 0 = OFF 1 = ON
1	0x0	READ: disable read from secure registers 0 = OFF 1 = ON
0	0x0	WRITE: disable write to secure registers 0 = OFF 1 = ON

9.1.15.3 APBDEV_PMC_PMC_SWRST_0

Reset only by por cell and sets itself back to inactive after 2 clock cycles. Only for emergency debugging purpose and not to be used in any functional mode.

Register write which resets PMC only

Offset: 008h | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
0	0x0	RST: software reset to pmc only 0 = DISABLE 1 = ENABLE

9.1.15.4 APBDEV_PMC_WAKE_MASK_0

Masks which event can cause wake-up from Deep Sleep mode. Has to be setup before entering Deep Sleep. Works in conjunction with WAKE_LVL register.

Only enabled events at the proper wake_lvl will cause exit from Deep Sleep mode.

Current pin assignments wake_mask, wake_status, sw_wake_status and wake levels:

- WAKE_STATUS[0] = ulpi_data4
- WAKE_STATUS[1] = gp3_pv[3]
- WAKE_STATUS[2] = dvi_d3
- WAKE_STATUS[3] = sdio3_dat1
- WAKE_STATUS[4] = hdmi_int
- WAKE_STATUS[5] = vgp[6]
- WAKE_STATUS[6] = gp3_pu[5]
- WAKE_STATUS[7] = gp3_pu[6]
- WAKE_STATUS[8] = gmi_wp_n
- WAKE_STATUS[9] = gp3_ps[2]
- WAKE_STATUS[10] = gmi_ad21
- WAKE_STATUS[11] = spi2_cs2
- WAKE_STATUS[12] = spi2_cs1
- WAKE_STATUS[13] = sdio1_dat1
- WAKE_STATUS[14] = gp3_pv[6]
- WAKE_STATUS[15] = gmi_ad16
- WAKE_STATUS[16] = rtc_irq
- WAKE_STATUS[17] = kbc_interrupt
- WAKE_STATUS[18] = pwr_int
- WAKE_STATUS[19] = usb_vbus_wakeup[0]
- WAKE_STATUS[20] = usb_vbus_wakeup[1]
- WAKE_STATUS[21] = usb_iddig[0]
- WAKE_STATUS[22] = usb_iddig[1]
- WAKE_STATUS[23] = gmi_iordy
- WAKE_STATUS[24] = gp3_pv[2]
- WAKE_STATUS[25] = gp3_ps[4]
- WAKE_STATUS[26] = gp3_ps[5]
- WAKE_STATUS[27] = gp3_ps[0]
- WAKE_STATUS[28] = gp3_pq[6]
- WAKE_STATUS[29] = gp3_pq[7]
- WAKE_STATUS[30] = dap1_dout
- WAKE_STATUS[31] = 0

PMC Wake-up Event Mask

Offset: 00ch | Read/Write: R/W | Reset: 0b00000000000000000000

Bit	Reset	Description
31	0x0	RESET_N: external reset wake enable 0 = DISABLE 1 = ENABLE
30:23	0x0	EVENT_RES: 0 = DISABLE 1 = ENABLE
22:19	0x0	USB_EVENT: 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
18	0x0	PWR_INT_N: PWR_INT_N wake enable 0 = DISABLE 1 = ENABLE
17	0x0	KBC: KBC wake enable 0 = DISABLE 1 = ENABLE
16	0x0	RTC: RTC wake enable 0 = DISABLE 1 = ENABLE
15:0	0x0	EVENT: pin 0-15 wake enable 0 = DISABLE 1 = ENABLE

9.1.15.5 APBDEV_PMC_WAKE_LVL_0

This register sets level which is active level for wake event. Will cause exit from Deep Sleep mode if input signal level matches level set in this register and WAKE_MASK is set for the event to 1

PMC Wake Level

Offset: 010h | Read/Write: R/W | Reset: 0b01111111100111111111111111111111

Bit	Reset	Description
31	0x0	RESET_N: external reset wake level (low active!) 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
30:23	0xff	EVENT_RES: 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
22:19	0x3	USB_EVENT: 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
18	0x1	PWR_INT_N: power interrupt - now permanently tied to bit 18 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
17	0x1	KBC: KBC wake level 0 = ACTIVE_LOW 1 = ACTIVE_HIGH

Bit	Reset	Description
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

9.1.15.6 APBDEV_PMC_WAKE_STATUS_0

This register stores status of the wake events. Event will be set if level matches and is not masked.

Events will be latched from the time when minimum power down timer expires (see scratch register 23) to first non-masked event being active. Due to 32 kHz granularity on the control, more than one event can become active before latching is disabled.

This register has valid content only after leaving Deep Sleep mode. Writing 1 (bitwise) resets status for the event.

PMC Wake Status

Offset: 014h | Read/Write: R/W | Reset: 0b00000000000000000000

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

9.1.15.7 APBDEV_PMC_SW_WAKE_STATUS_0

This register stores status of the wake events. Latching of the events in software wake status is enabled by PMC_CNRL register bit LATCHWAKE_EN.

Latching will stop at 1 to 0 transition on this bit. Event will be set if level matches. Masking is not affecting this register. Write will reset wake events set.

PMC Software Wake Status

Offset: 018h | Read/Write: R/W | Reset: 0b00000000000000000000

Bit	Reset	Description
31	0x0	RESET_N: external reset 0 = NOT_SET 1 = SET
30:23	0x0	EVENT_RES: 0 = NOT_SET 1 = SET
22:19	0x0	USB_EVENT: USB wake events 0 = NOT_SET 1 = SET
18	0x0	PWR_INT: power interrupt 0 = NOT_SET 1 = SET
17	0x0	KBC: KBC wake 0 = DISABLE 1 = ENABLE
16	0x0	RTC: RTC wake 0 = DISABLE 1 = ENABLE
15:0	0x0	EVENT: pin 0-15 wake 0 = DISABLE 1 = ENABLE

9.1.15.8 APBDEV_PMC_DPD_PADS_ORIDE_0

This register enables overriding values from pinmux with values driven by KBC (keyboard) or PMC (sys_clk_req, blink).

If bit is set to 1, particular IO will drive direct value from KBC or PMC, and not the pinmux value. During LP0 mode, the pads with the bit set to 1 will not be in deep power down mode. The IO will not drive data from mini pad macros stored during sample cycle, but direct data from KBC or PMC.

Note: This register has to be set before entering LP0 mode.

DPD Pads Override

Offset: 01ch | Read/Write: R/W | Reset: 0b00001000000000000000000000000000

Bit	Reset	Description
25	0x0	KBC_COL7: override dpd idle state with column 7 output 0 = DISABLE 1 = ENABLE
24	0x0	KBC_COL6: override dpd idle state with column 6 output 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
23	0x0	KBC_COL5: override dpd idle state with column 5 output 0 = DISABLE 1 = ENABLE
22	0x0	KBC_COL4: override dpd idle state with column 4 output 0 = DISABLE 1 = ENABLE
21	0x1	SYS_CLK: override dpd idle state with column with sys_clk_request output 0 = DISABLE 1 = ENABLE
20	0x0	BLINK: override dpd idle state with blink ouptut 0 = DISABLE 1 = ENABLE
19	0x0	KBC_COL3: override dpd idle state with column 3 output 0 = DISABLE 1 = ENABLE
18	0x0	KBC_COL2: override dpd idle state with column 2 output 0 = DISABLE 1 = ENABLE
17	0x0	KBC_COL1: override dpd idle state with column 1 output 0 = DISABLE 1 = ENABLE
16	0x0	KBC_COL0: override dpd idle state with column 0 output 0 = DISABLE 1 = ENABLE
15	0x0	KBC_ROW15: override dpd idle state with row 15 output 0 = DISABLE 1 = ENABLE
14	0x0	KBC_ROW14: override dpd idle state with row 14 output 0 = DISABLE 1 = ENABLE
13	0x0	KBC_ROW13: override dpd idle state with row 13 output 0 = DISABLE 1 = ENABLE
12	0x0	KBC_ ROW 12: override dpd idle state with row 12 output 0 = DISABLE 1 = ENABLE
11	0x0	KBC_ ROW 11: override dpd idle state with row 11 output 0 = DISABLE 1 = ENABLE
10	0x0	KBC_ ROW 10: override dpd idle state with row 10 output 0 = DISABLE 1 = ENABLE
9	0x0	KBC_ ROW 9: override dpd idle state with row 9 output 0 = DISABLE 1 = ENABLE
8	0x0	KBC_ ROW 8: override dpd idle state with row 8 output 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
7	0x0	KBC_ ROW 7: override dpd idle state with row 7 output 0 = DISABLE 1 = ENABLE
6	0x0	KBC_ ROW 6: override dpd idle state with row 6 output 0 = DISABLE 1 = ENABLE
5	0x0	KBC_ ROW 5: override dpd idle state with row 5 output 0 = DISABLE 1 = ENABLE
4	0x0	KBC_ ROW 4: override dpd idle state with row 4 output 0 = DISABLE 1 = ENABLE
3	0x0	KBC_ ROW 3: override dpd idle state with row 3 output 0 = DISABLE 1 = ENABLE
2	0x0	KBC_ ROW 2: override dpd idle state with row 2 output 0 = DISABLE 1 = ENABLE
1	0x0	KBC_ ROW 1: override dpd idle state with row 1 output 0 = DISABLE 1 = ENABLE
0	0x0	KBC_ROW0: override dpd idle state with row 0 output 0 = DISABLE 1 = ENABLE

9.1.15.9 APBDEV_PMC_DPD_SAMPLE_0

Setting this register will trigger sampling pads data and direction in which pad will be driven during Deep Sleep mode.

Before writing to this register, all interfaces going to pads must be set to ideal "idle" mode, which is expected to be driven by pads when chip enters Deep Sleep.

DPS Power Down Sample has to precede Deep Power Down Enable write. DPD Sample shouldn't be de-asserted until transition from Deep Sleep to WB0.

Deep Power Down Sample

Offset: 020h | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
0	0x0	ON: will set sampling of pads value 0 = DISABLE 1 = ENABLE

9.1.15.10 APBDEV_PMC_DPD_ENABLE_0

Setting this register will trigger entering Deep Sleep state. Must be preceded by DPD_SAMPLE write.

Entering deep power down mode (Deep Sleep) will trigger shut down core power request and shut down system clock request. PLLs, IOs will enter power deep down mode. All signals going from core power to RTC and pads will be cut off (clamped)

If none of the wake-up events is set, only power on reset can enable back access to the chip.

After serving wake-up event is completed, the register should be set back to 0 to complete Deep Sleep cycle.

Deed Power Down Enable

Offset: 024h | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
0	0x0	ON: will set sampling of pads value 0 = DISABLE 1 = ENABLE

9.1.15.11 APBDEV_PMC_PWRGATE_TIMER_OFF_0

Specifies number of APB cycles after which the rail line goes off (turns the power to part of power gated partition). Each rail controls part of powered partition. This register should be set before write to Power Gate Toggle. Shared between all power partitions.

Power Gate Timer Register

Offset: 028h | Read/Write: R/W | Reset: 0b11101100101010010111010100110001

Bit	Reset	Description
31:28	0xe	RAIL7: timer value for rail 7
27:24	0xc	RAIL6: timer value for rail 6
23:20	0xa	RAIL5: timer value for rail 5
19:16	0x9	RAIL4: timer value for rail 4
15:12	0x7	RAIL3: timer value for rail 3
11:8	0x5	RAIL2: timer value for rail 2
7:4	0x3	RAIL1: timer value for rail 1
3:0	0x1	RAIL0: timer value for rail 0

9.1.15.12 APBDEV_PMC_PWRGATE_TIMER_ON_0

Specifies number of APB cycles after which the rail line goes on. Shared between all power partitions. Has to be reloaded before each power gate on/off command.

Power Gate Timer Register

Offset: 02ch | Read/Write: R/W | Reset: 0b11101100101010010111010100110001

Bit	Reset	Description
31:28	0xe	RAIL7: timer value for rail 7
27:24	0xc	RAIL6: timer value for rail 6
23:20	0xa	RAIL5: timer value for rail 5
19:16	0x9	RAIL4: timer value for rail 4
15:12	0x7	RAIL3: timer value for rail 3

Bit	Reset	Description
11:8	0x5	RAIL2: timer value for rail 2
7:4	0x3	RAIL1: timer value for rail 1
3:0	0x1	RAIL0: timer value for rail 0

9.1.15.13 APBDEV_PMC_PWRGATE_TOGGLE_0

Write to this register will turn power on/off on to specified power gated partition. Only one partition will be turned on/off at the time.

PWRGATE_STATUS should be read to determine state of the partition before writing to this register.

Turning partition off will cause automatic clamping of all signals generated by the power gated partition being turned off. Before partition is turned off, all clocks to the partition should be stopped and reset to the partition should be asserted.

Turning partition on will not remove clamping. Clamping will be removed only after REMOVE_CLAMPING_CMD write.

Power Gate Toggle

Offset: 030h | Read/Write: R/W | Reset: 0b0xxxxx000

Bit	Reset	Description
8	0x0	START: start power down/up 0 = DISABLE 1 = ENABLE
2:0	0x0	PARTID: id of partition to be toggled 0 = CPU Partition 1 = TD (3D) 2 = Video Encode 4 = Video Decode 3 = PCI Express 5 = L2 Cache 6 = MPEG Encode

9.1.15.14 APBDEV_PMC_REMOVE_CLAMPING_CMD_0

This is a bit map with one bit per power partition controller by PMC.

When written to 1, the PMC removes the clamp signals to the corresponding partition. If the partition is not powered on, the register write will be ignored and clamping of the partition will continue

The bit is automatically reset to 0 when the clamping has been removed. SW is responsible to write to this register at a correct time, that is, when the clocks are started but the blocks in that partition are still held in reset.

Remove Clamping

Offset: 034h | Read/Write: R/W | Reset: 0b0000000

Bit	Reset	Description
6	0x0	MPE: remove clamping to MPE_CACHE 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
5	0x0	L2C: remove clamping to L2_CACHE 0 = DISABLE 1 = ENABLE
4	0x0	PCX: remove clamping to PCX 0 = DISABLE 1 = ENABLE
3	0x0	VDE: remove clamping to VDE 0 = DISABLE 1 = ENABLE
2	0x0	VE: remove clamping to VE 0 = DISABLE 1 = ENABLE
1	0x0	TD: remove clamping to TD 0 = DISABLE 1 = ENABLE
0	0x0	CPU: remove clamping to CPU 0 = DISABLE 1 = ENABLE

9.1.15.15 APBDEV_PMC_PWRGATE_STATUS_0

Displays status of the power partitions. This register should be read before writing to PWRGATE_TOGGLE. Read only register.

Power Gate Status

Offset: 038h | Read/Write: RO | Reset: 0bxxxxxxx

Bit	Reset	Description
6	X	MPE: status of MPE partition 0 = OFF 1 = ON
5	X	L2C: status of L2C partition 0 = OFF 1 = ON
4	X	VDE: status of VDE partition 0 = OFF 1 = ON
3	X	PCX: status of PCX partition 0 = OFF 1 = ON
2	X	VE: status of VE partition 0 = OFF 1 = ON
1	X	TD: status of TD Partition 0 = OFF 1 = ON

Bit	Reset	Description
0	X	CPU: status of CPU partition 0 = OFF 1 = ON

9.1.15.16 APBDEV_PMC_PWRGOOD_TIMER_0

There are two time periods controlled by this register:

- Power Stabilization Time from Core Power Request – the time required for the PMIC to provide a stable Core Power Rail.
- Oscillator Stabilization Time – the time required to ensure that an external oscillator or the internal oscillator is stabilized prior to PMC de-asserting the internal reset signal as the processor resumes from the Deep Sleep state.

Power Good Timer

Offset: 03ch | Read/Write: R/W | Reset: 0b0000000001111111

Bit	Reset	Description
15:8	0x0	Oscillator Stabilization Time (<Value 15:8>*16*0.0351) – (<Value 7:0> * 0.03051)m
7:0	0x7f	Power Stabilization Time from Core Power Request (<Value 7:0> * 0.03051)mS

9.1.15.17 APBDEV_PMC_BLINK_TIMER_0

Blinker Timer, for external blinker Will output value to pad only if 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<<4 x 30.51 us
- Off time DATA_OFF<<4 x 30.51 us.

Blinker Timer

Offset: 040h | Read/Write: R/W | Reset: 0b11111111111111111111111111111111

Bit	Reset	Description
31:16	0xffff	DATA_OFF: time off
15	0x1	FORCE_BLINK: if 0 32khz clock
14:0	0xffff	DATA_ON: time on

9.1.15.18 APBDEV_PMC_NO_IOPOWER_0

Controls power rails for each IO level. Can only be set when specific power rail is off. Setting it otherwise will damage the pad.

No IO Power Register

Offset: 044h | Read/Write: R/W | Reset: 0b0000000000

Bit	Reset	Description

Bit	Reset	Description
9	0x0	DSI_CSI: rail dsi_csi IOs 0 = DISABLE 1 = ENABLE
8	0x0	SD: rail sd IOs 0 = DISABLE 1 = ENABLE
7	0x0	MEM: rail mem IOs 0 = DISABLE 1 = ENABLE
6	0x0	LCD: rail lcd IOs 0 = DISABLE 1 = ENABLE
5	0x0	AUDIO: rail i2s IOs 0 = DISABLE 1 = ENABLE
4	0x0	VI: rail dvi IOs 0 = DISABLE 1 = ENABLE
3	0x0	BB: dlcd IOs 0 = DISABLE 1 = ENABLE
2	0x0	UART: rail dbg IOs 0 = DISABLE 1 = ENABLE
1	0x0	NAND: rail at3 IOs 0 = DISABLE 1 = ENABLE
0	0x0	SYS: rail ao IOs 0 = DISABLE 1 = ENABLE

9.1.15.19 APBDEV_PMC_PWR_DET_0

Power detect cells identify rails that are higher than 2.5V to ensure IO cells are properly configured for these interfaces. After an IO interface rail is powered, the power detect cell for that rail must be enabled and the power detect output latched before the interface is enabled for use by Tegra 2 Series devices.

After the power detect output has been latched, the power detect cell should be disabled to reduce idle power for Tegra 2 Series devices.

Active high, sets power detection, only for nine power rails - DSI_CSI doesn't have power detect.

The proper sequence for turning on power detects:

- write 1 to PWR_DET register for fields requiring power detect
- Allow for 3 us delay (in abp clock cycles)
- write 1 to PWR_DET_LATCH

For turning PWR_DET off:

- write 0 to PWR_DET_LATCH
- no delay necessary
- write 0 to PWR_DET

Power Detect

Offset: 048h | Read/Write: R/W | Reset: 0b111111111

Bit	Reset	Description
8	0x1	SD: rail sd IOs 0 = ENABLE 1 = DISABLE
7	0x1	MEM: rail mem IOs 0 = ENABLE 1 = DISABLE
6	0x1	LCD: rail lcd IOs 0 = ENABLE 1 = DISABLE
5	0x1	AUDIO: rail i2s IOs 0 = ENABLE 1 = DISABLE
4	0x1	VI: rail dvi IOs 0 = ENABLE 1 = DISABLE
3	0x1	BB: rail dlcd IOs 0 = ENABLE 1 = DISABLE
2	0x1	UART: rail dbg IOs 0 = ENABLE 1 = DISABLE
1	0x1	NAND: rail at3 IOs 0 = ENABLE 1 = DISABLE
0	0x1	SYS: rail ao IOs 0 = ENABLE 1 = DISABLE

9.1.15.20 APBDEV_PMC_PWR_DET_LATCH_0

Latches power detect for power rails enabled by Power Detect register.

Power Detect Latch

Offset: 04ch | Read/Write: R/W | Reset: 0b1

Bit	Reset	Description
0	0x1	LATCH: power detect latch, latches value from the pads as long set to 1 0 = ENABLE 1 = DISABLE

9.1.15.21 APBDEV_PMC_SCRATCH0_0

Scratch Register

Scratch registers for restoring context after wake-up. On a cold power up, the content of reg 0 will be reset to 0x0.

Offset: 050h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	SCRATCH0: General purpose register storage

9.1.15.22 APBDEV_PMC_SCRATCH1_0

Scratch Register

Offset: 054h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH1: General purpose register storage

9.1.15.23 APBDEV_PMC_SCRATCH2_0

Scratch Register

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

Bit	Reset	Description
31:0	X	SCRATCH2: General purpose register storage

9.1.15.24 APBDEV_PMC_SCRATCH3_0

Scratch Register

Offset: 05ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH3: General purpose register storage

9.1.15.25 APBDEV_PMC_SCRATCH4_0

Scratch Register

Offset: 060h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH4: General purpose register storage

9.1.15.26 APBDEV_PMC_SCRATCH5_0

Scratch Register

Offset: 064h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH5: General purpose register storage

9.1.15.27 APBDEV_PMC_SCRATCH6_0

Scratch Register

Offset: 068h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH6: General purpose register storage

9.1.15.28 APBDEV_PMC_SCRATCH7_0

Scratch Register

Offset: 06ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH7: General purpose register storage

9.1.15.29 APBDEV_PMC_SCRATCH8_0

Scratch Register

Offset: 070h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH8: General purpose register storage

9.1.15.30 APBDEV_PMC_SCRATCH9_0

Scratch Register

Offset: 074h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH9: General purpose register storage

9.1.15.31 APBDEV_PMC_SCRATCH10_0

Scratch Register

Offset: 078h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH10: General purpose register storage

9.1.15.32 APBDEV_PMC_SCRATCH11_0

Scratch Register

Offset: 07ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH11: General purpose register storage

9.1.15.33 APBDEV_PMC_SCRATCH12_0

Scratch Register

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

Bit	Reset	Description
31:0	X	SCRATCH12: General purpose register storage

9.1.15.34 APBDEV_PMC_SCRATCH13_0

Scratch Register

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

Bit	Reset	Description
31:0	X	SCRATCH13: General purpose register storage

9.1.15.35 APBDEV_PMC_SCRATCH14_0

Scratch Register

Offset: 088h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH14: General purpose register storage

9.1.15.36 APBDEV_PMC_SCRATCH15_0

Scratch Register

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

Bit	Reset	Description
31:0	X	SCRATCH15: General purpose register storage

9.1.15.37 APBDEV_PMC_SCRATCH16_0

Scratch Register

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

Bit	Reset	Description
31:0	X	SCRATCH16: General purpose register storage

9.1.15.38 APBDEV_PMC_SCRATCH17_0

Scratch Register

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

Bit	Reset	Description
31:0	X	SCRATCH17: General purpose register storage

9.1.15.39 APBDEV_PMC_SCRATCH18_0

Scratch Register

Offset: 098h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH18: General purpose register storage

9.1.15.40 APBDEV_PMC_SCRATCH19_0

Scratch Register

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

Bit	Reset	Description
31:0	X	SCRATCH19: General purpose register storage

9.1.15.41 APBDEV_PMC_SCRATCH20_0

Scratch Register

Offset: 0a0h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH20: General purpose register storage

9.1.15.42 APBDEV_PMC_SCRATCH21_0

Scratch Register

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

Bit	Reset	Description
31:0	X	SCRATCH21: General purpose register storage

9.1.15.43 APBDEV_PMC_SCRATCH22_0

Scratch Register

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

Bit	Reset	Description
31:0	X	SCRATCH22: General purpose register storage

9.1.15.44 APBDEV_PMC_SCRATCH23_0

Scratch Register

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

Bit	Reset	Description
31:0	X	SCRATCH23: General purpose register storage

9.1.15.45 APBDEV_PMC_SECURE_SCRATCH0_0

Secure Scratch Register

Offset: 0b0h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SECURE_SCRATCH0

9.1.15.46 APBDEV_PMC_SECURE_SCRATCH1_0

Secure Scratch Register

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

Bit	Reset	Description
31:0	X	SECURE_SCRATCH1

9.1.15.47 APBDEV_PMC_SECURE_SCRATCH2_0

Secure Scratch Register

Offset: 0b8h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SECURE_SCRATCH2

9.1.15.48 APBDEV_PMC_SECURE_SCRATCH3_0

Secure Scratch Register

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

Bit	Reset	Description
31:0	X	SECURE_SCRATCH3

9.1.15.49 APBDEV_PMC_SECURE_SCRATCH4_0

Secure Scratch Register

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

Bit	Reset	Description
31:0	X	SECURE_SCRATCH4

9.1.15.50 APBDEV_PMC_SECURE_SCRATCH5_0

Secure Scratch Register

Offset: 0c4h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SECURE_SCRATCH5

9.1.15.51 APBDEV_PMC_CPUPWRGOOD_TIMER_0

CPU Power Good Timer

SW 33v override - will work only if SW_E33V_OVR_EN is enabled. 1 will set value to 33v, 0 to 2.5

WARNING! Setting 0 on 3.3v interface will damage the chip!

Each rail has enable bit, only when enable is set, the value is written.

The behavior for timer DATA of zero and one will be the same (i.e., 1 cycle of delay)

Offset: 0c8h | Read/Write: R/W | Reset: 0b00000000000000001111111111111111

Bit	Reset	Description
31:0	0xffff	DATA: timer data

9.1.15.52 APBDEV_PMC_CPUPWROFF_TIMER_0

CPU Power Off Timer

Used for On to Off transition.

Offset: 0cch | Read/Write: R/W | Reset: 0b00000000000000001111111111111111

Bit	Reset	Description
31:0	0xffff	DATA: timer data

9.1.15.53 APBDEV_PMC_PG_MASK_0

Offset: 0d0h | Read/Write: R/W | Reset: 0b11111111111111111111111111111111

Bit	Reset	Description
31:24	0xff	PX: Mask PCX rail
23:16	0xff	VD: Mask VDE rail
15:8	0xff	VE: Mask VE rail
7:0	0xff	TD: Mask TD rail

9.1.15.54 APBDEV_PMC_PG_MASK_1_0

Offset: 0d4h | Read/Write: R/W | Reset: 0b11111111xxxxxxx1

Bit	Reset	Description
15:8	0xff	MPE: MASK MPE rail
0	0x1	L2C: MASK L2C rail

9.1.15.55 APBDEV_PMC_AUTO_WAKE_LVL_0

> **Note:** This register should not be programmed.

The wake levels are snapped just before entering DPD by default.

Offset: 0d8h | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
0	0x0	SMPL: Causes PMC to sample the wake pads 0 = DISABLE 1 = ENABLE

9.1.15.56 APBDEV_PMC_AUTO_WAKE_LVL_MASK_0

This register is used by s/w to enable sampling of the wake pads before entering LP0

Setting a '1' would cause the associated pad to be sampled and value transferred to WAKE_LVL register

Offset: 0dch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	VALUE

9.1.15.57 APBDEV_PMC_WAKE_DELAY_0

Offset: 0e0h | Read/Write: R/W | Reset: 0b0000000000000000

Bit	Reset	Description
15:0	0x0	VALUE

9.1.15.58 APBDEV_PMC_PWR_DET_VAL_0

This register is used to override the power-detect cells and manually set the values (in unlikely case the power-detect cells are broken). A write to this register also causes the values from the power-detect cells to be captured and which can be subsequently read out.

Offset: 0e4h | Read/Write: R/W | Reset: 0b111111111

Bit	Reset	Description
8	0x1	SD: rail sd IOs 0 = ENABLE 1 = DISABLE
7	0x1	MEM: rail mem IOs 0 = ENABLE 1 = DISABLE
6	0x1	LCD: rail lcd IOs 0 = ENABLE 1 = DISABLE
5	0x1	AUDIO: rail i2s IOs 0 = ENABLE 1 = DISABLE
4	0x1	VI: rail dvi IOs 0 = ENABLE 1 = DISABLE
3	0x1	BB: rail dlcd IOs 0 = ENABLE 1 = DISABLE
2	0x1	UART: rail dbg IOs 0 = ENABLE 1 = DISABLE
1	0x1	NAND: rail at3 IOs 0 = ENABLE 1 = DISABLE
0	0x1	SYS: rail ao IOs 0 = ENABLE 1 = DISABLE

9.1.15.59 APBDEV_PMC_DDR_PWR_0

This register is used to program the "E_18V" pin of the ddr pads.

Offset: 0e8h | Read/Write: R/W | Reset: 0b1

Bit	Reset	Description
0	0x1	VAL: 0 = E_12V 1 = E_18V

9.1.15.60 APBDEV_PMC_USB_DEBOUNCE_DEL_0

Offset: 0ech | Read/Write: R/W | Reset: 0b0000000000000000

Bit	Reset	Description
15:0	0x0	VAL

9.1.15.61 APBDEV_PMC_USB_AO_0

Offset: 0f0h | Read/Write: R/W | Reset: 0b0000

Bit	Reset	Description
3:2	0x0	VBUS_WAKEUP_PD
1:0	0x0	UB_ID_PD

9.1.15.62 APBDEV_PMC_CRYPTO_OP_0

A complete solution requires a "semi-sticky" bit in the always-on domain. The Boot ROM would clear this semi-sticky bit for cold boots, but use its value to propagate the crypto-disable flag across LP0. (The Boot ROM always requires crypto functionality, so a pure hardware solution probably isn't reasonable.)

The Boot ROM would be able to clear (to zero) and read the sticky bit, but outside the Boot ROM users would only be able to set (to one) and read the sticky bit. On cold boot, the Boot ROM would always clear the stick bit. On WB0, the Boot ROM would read the sticky bit and copy it's setting to the crypto disable flag.

Crypto Engine Disable Sticky Bit

Offset: 0f4h | Read/Write: R/W | Reset: 0b1

Bit	Reset	Description
0	0x1	VAL: Disabled by default 0 = ENABLE 1 = DISABLE

9.1.15.63 APBDEV_PMC_PLLP_WB0_OVERRIDE_0

Offset: 0f8h | Read/Write: R/W | Reset: 0b0000

Bit	Reset	Description
3:2	0x0	OSC_FREQ: 00 = 13MHz, 01 = 19.2MHz, 10 = 12MHz, 11 = 26MHz
1	0x0	PLLP_ENABLE: 1 = enable PLLP, 0 = disable PLLP
3:0	0x0	VAL
0	0x0	OVERRIDE_ENABLE: 1 = override CAR PLLP setting, 0 = no override.

9.1.15.64 APBDEV_PMC_SCRATCH24_0

Scratch Register

Offset: 0fch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH24: General purpose register storage

9.1.15.65 APBDEV_PMC_SCRATCH25_0

Scratch Register

Offset: 100h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH25: General purpose register storage

9.1.15.66 APBDEV_PMC_SCRATCH26_0

Scratch Register

Offset: 104h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH26: General purpose register storage

9.1.15.67 APBDEV_PMC_SCRATCH27_0

Scratch Register

Offset: 108h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH27: General purpose register storage

9.1.15.68 APBDEV_PMC_SCRATCH28_0

Scratch Register

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

Bit	Reset	Description
31:0	X	SCRATCH28: General purpose register storage

9.1.15.69 APBDEV_PMC_SCRATCH29_0

Scratch Register

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

Bit	Reset	Description
31:0	X	SCRATCH29: General purpose register storage

9.1.15.70 APBDEV_PMC_SCRATCH30_0

Scratch Register

Offset: 114h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH30: General purpose register storage

9.1.15.71 APBDEV_PMC_SCRATCH31_0

Scratch Register

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

Bit	Reset	Description
31:0	X	SCRATCH31: General purpose register storage

9.1.15.72 APBDEV_PMC_SCRATCH32_0

Scratch Register

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

Bit	Reset	Description
31:0	X	SCRATCH32: General purpose register storage

9.1.15.73 APBDEV_PMC_SCRATCH33_0

Scratch Register

Offset: 120h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH33: General purpose register storage

9.1.15.74 APBDEV_PMC_SCRATCH34_0

Scratch Register

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

Bit	Reset	Description
31:0	X	SCRATCH34: General purpose register storage

9.1.15.75 APBDEV_PMC_SCRATCH35_0

Scratch Register

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

Bit	Reset	Description
31:0	X	SCRATCH35: General purpose register storage

9.1.15.76 APBDEV_PMC_SCRATCH36_0

Scratch Register

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

Bit	Reset	Description
31:0	X	SCRATCH36: General purpose register storage

9.1.15.77 APBDEV_PMC_SCRATCH37_0

Scratch Register

Offset: 130h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH37: General purpose register storage

9.1.15.78 APBDEV_PMC_SCRATCH38_0

Scratch Register

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

Bit	Reset	Description
31:0	X	SCRATCH38: General purpose register storage

9.1.15.79 APBDEV_PMC_SCRATCH39_0

Scratch Register

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

Bit	Reset	Description
31:0	X	SCRATCH39: General purpose register storage

9.1.15.80 APBDEV_PMC_SCRATCH40_0

Scratch Register

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

Bit	Reset	Description
31:0	X	SCRATCH40: General purpose register storage

9.1.15.81 APBDEV_PMC_SCRATCH41_0

Scratch Register

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

Bit	Reset	Description
31:0	X	SCRATCH41: General purpose register storage

9.1.15.82 APBDEV_PMC_SCRATCH42_0

Scratch Register

Bit 0 of this register is tied to PCX_CLAMP.

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

Bit	Reset	Description
31:0	X	SCRATCH42: General purpose register storage

9.1.15.83 APBDEV_PMC_BONDOUT_MIRROR0_0

Secure Scratch Register

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

Bit	Reset	Description
31:0	X	BONDOUT_MIRROR0

9.1.15.84 APBDEV_PMC_BONDOUT_MIRROR1_0

Secure Scratch Register

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

Bit	Reset	Description
31:0	X	BONDOUT_MIRROR1

9.1.15.85 APBDEV_PMC_BONDOUT_MIRROR2_0

Secure Scratch Register

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

Bit	Reset	Description
31:0	X	BONDOUT_MIRROR2

9.1.15.86 APBDEV_PMC_SYS_33V_EN_0

Offset: 154h | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
0	0x0	VAL: 1 = 3.3v 0 = 1.8v

9.1.15.87 APBDEV_PMC_BONDOUT_MIRROR_ACCESS_0

On separate reset (same as car) disables access (read/write to secure scratch registers)

Offset: 158h | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
1	0x0	BREAD: disable read from bondout secure registers 0 = OFF 1 = ON
0	0x0	BWRITE: disable write to bondout secure registers 0 = OFF 1 = ON

9.1.15.88 APBDEV_PMC_GATE_0

This register is used for SW controlled synchronization between APB domain and the 32 KHz domain. SW is expected to set the GAKE_WAKE/GATE_DBNS field high to cut the 32 KHz gated clock before updating WAKE_LVL, AUTO_WAKE_LVL, WAKE_MASK and USB_DEBOUNCE_DEL registers. After these registers have been written the GATE_WAKE/GATE_DBNS bit should be written back to '0' to enabled the 32 khz gated clock.

This gates the 32 KHz clock to just the flops that store the above fields.

Offset: 15ch | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
1	0x0	GATE_WAKE: 0 = OFF 1 = ON
0	0x0	GATE_DBNS: 0 = OFF 1 = ON

9.2 Dynamic Voltage Controller

Dynamic Voltage Controller (DVC) can optionally be used as part of the DVFS sub-system. DVFS sub-system provides features that allow the chip to be operated at the lowest voltage and frequency for a given application load thereby reducing the power consumption.

The DVC implements two high-level functions:

- An I2C Master Controller. For details on the functionality and programming guidelines of the I2C Master Controller, please refer to the section on I2C Controller in this document.
- Hardware to measure silicon performance and derive values to program the voltage of an external PMU.

The Tegra® 2 processors are designed to meet worst case performance requirements based on the worst case load that a set of application use cases may present. While running applications that do not require the full performance of the Tegra 2, the frequency and voltage can be lowered, reducing the power dissipation and enhancing battery life.

The minimum frequency required to support a particular use case can be determined by adding the MIPS/MHz requirement of all the applications that are running. The minimum voltage required to keep the device functional at a particular frequency and application load can be determined by lowering the voltage up to a point where the application runs without failures. Determining minimum frequency and voltage requires extensive characterization of silicon under various PVT (Process, Voltage & Temperature) conditions.

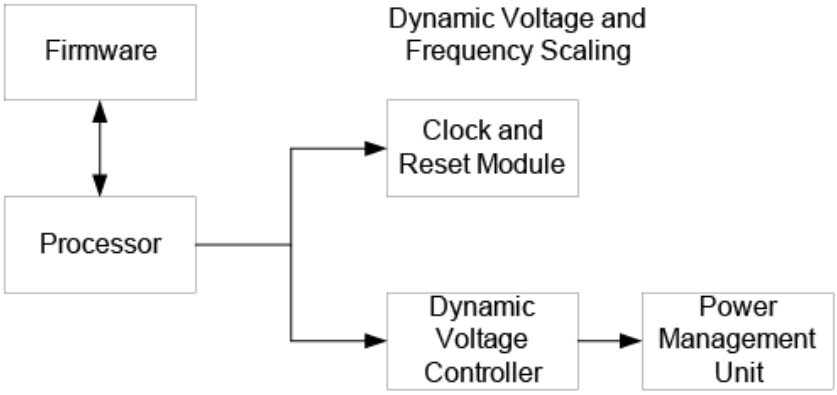
Hardware Features

The DVC controller implements:

- Two ring oscillators to measure performance of silicon under various PVT conditions:
  - A 32-bit adder which adds '-1' and '0' with one of sum0 through sum31 being looped back
  - A “Speedo” ring oscillator
- A mux to select long RC path in the feedback loop.
- A 64-entry Voltage Lookup Table (VLUT) that stores the tuple: Minimum Voltage, Target Performance Count, Maximum or Safe Voltage.
- A firmware initiated voltage adjustment.
- An I2C master interface that can be used by FSM to program the appropriate voltage select value to external PMU. The I2C master interface also provides a driver code interface so that driver code can directly program the external PMU.
- I2C controller also supports packet mode , which allows large amount of data transfers with repeat start option
- A programmable counter for periodic voltage tuning

9.2.1 DVFS Overview

Figure 15. Dynamic Voltage and Frequency Scaling



DVFS, of which DVC is a part, is a sub-system that allows the device to operate at lowest possible frequency and voltage for a given application load. Firmware running on the processor programs the desired frequency in clock-and-reset module.

9.2.2 Operation

Firmware, upon request to launch or kill an application, computes the application load and the minimum frequency required to run the application(s). Firmware normalizes the frequency and then triggers the DVC module by programming the request register. DVC in turn looks up the VLUT to determine the voltage select value corresponding to the safe operating voltage for that target performance value. DVC then programs the external PMU via I2C interface, waits for voltage to stabilize and interrupts firmware. Firmware then programs the registers in CAR to switch the PLL frequency or clock source.

Each step described above can acknowledge immediately, thereby reducing the loop delay, if the application functionality is guaranteed. For example, when changing to a lower voltage, DVC can acknowledge immediately upon receiving the trigger. It is also possible that some steps can be done in parallel.

For example, PLL stabilization and PMU voltage ramp up can be done in parallel assuming that there is a net reduction in loop delay as compared to the case where the steps are performed in sequence.

DVC controls the voltage rail it is connected to, which in the Tegra 2 devices is the MAIN power partition.

9.2.3 Voltage Lookup Table - VLUT

VLUT stores three values in each entry: maximum voltage or safe voltage (Vmax), minimum voltage (Vmin), performance count. DVC maintains the voltage within Vmax and Vmin and tries to achieve the performance count. The Vmax and Vmin that are stored in the table are actually voltage select values that need to be written to the PMU.

9.2.3.1 Maximum (Safe) Voltage

This is the voltage that guarantees operation at a particular frequency regardless of the temperature. Since it is difficult to determine with accuracy, the PVT of a particular device, some amount of margin is built in to this value. The margin that is built in to the safe voltage may be minimal and depends on how well the characterized data matches the measured performance during calibration.

9.2.3.2 Minimum Voltage

This is voltage below which device operation is not guaranteed for a particular frequency of operation. This is a safety mechanism and may not be needed. Setting this to a very low value will effectively disable this feature.

9.2.3.3 Performance Count

Performance count is the value below which performance of the device does not guarantee sustaining the load. This value is compared against the ring oscillator count and voltage is lowered if necessary.

9.2.3.4 Normalized Frequency

Normalized (scaled) frequency is used to index in to the VLUT. Frequency is normalized because the range of values it can assume have to be mapped to the size of VLUT array (64-entries). Software computes this value and writes to request register in DVC. It may be difficult to arrive at this number because each clock domain the chip could be operating at different frequency.

9.2.3.5 Characterization

As part of silicon bring up, devices are characterized across various PVT conditions and the performance counter values are saved in a set of tables. At each process corner, the following steps are performed to generate a table:

1. Set the frequency to a desired value
2. Set the ambient temperature to the highest value supported
3. Set the voltage to highest value supported
4. Run all the application use cases that can work at the above frequency
5. Lower the voltage until a failure is seen
6. Read the performance count
7. Repeat steps 2-5 for lowest supported ambient temperature
8. Repeat steps 1-6 for all supported frequencies

The voltage determined in Step 5 is the Vmax, maximum (safe) voltage, the voltage determined in Step 7 is the Vmin and the performance count is determined by taking the max of the values read in Step 6. The values need to be padded to account for inaccuracies in performance count as well as voltage fluctuations in the core. The above steps are repeated for all supported frequencies and application loads. It is possible that multiple tables are generated for each process corner based on the application use cases.

During boot, firmware reads the fuse register to determine the process corner, looks up the table that best matches the process corner and application use case and downloads it to VLUT.

Tegra 2 devices have multiple clock domains with the logic in each domain optimized to the maximum frequency that needs to be supported. Path delays increase when voltage is lowered which reduces the frequency of operation of a particular logic block. When voltage is lowered, care has to be taken so that paths in all clock domains do not fail. Because of this it might be necessary to generate tables for each clock domain for each process corner. Depending on the application in use and the frequency of the operation of each domain, a suitable can be selected to be downloaded to the VLUT.

Firmware needs to map Vmax and Vmin values in the tables to appropriate voltage select values required by the PMU that is used in the hardware configuration of interest and also needs to scale the performance count values appropriately depending on the reference clock frequency of the system.

9.2.4 I2C Packet Mode

DVC-I2C controller supports I2C packet mode, which removes constraint on the size of data transfer in normal mode. This allows fairly large number of data transfers with repeat start option. Multiple packets can be chained together and APBDMA support has been added for DVC-I2C. A packet consists of header and data. Header consists of config data such as length, slave address etc, while data part consists of actual data to be written. For more information on packet based interface functionality and registers, refer to the I2C section of this manual.

9.2.5 Restrictions

DVC relies on the following assumptions:

- Fuse module stores the process corner information.
- Ring oscillators provide good correlation to PVT variations.
- Measured silicon performance tracks the behavior of silicon speed paths, spread across the chip, accurately.
- Characterization data and subsequent analysis of the data provides a set of tables and an algorithm to select one of them for a particular device.
- Calibration results in an unambiguous conservative choice that can be made with regards to the values to be loaded in to VLUT of a particular device.
- External PMU does not require programming more than two registers to change the voltage.
- Total amount of delay it takes to change voltage is not significant enough to affect the functionality of the system.
- DVC is connected to the power grid that it is controlling the voltage for.
- The voltage select values are monotonically increasing for a monotonically increasing output voltage of the PMU.

9.2.6 Programming Guidelines

9.2.6.1 Use Case 1

DVC gets a request from driver code for voltage change request and DVC controls the External PMU via I2C. The following are the requirements.

- DVC should be in Continuous adjustment mode
- PMU settling time is 64 reference clocks
- Number of iterations is 4
- Interrupt is enabled
- Wakeup timer is disabled
- Ring oscillator should be sampled for 10 ref clocks (ROSC_SA_CNT)
- Ring oscillator settling time (ROSC_STRAT_DEL) is 2 ref clocks
- Firmware should know when DVC has completed the change request
- Hysteresis counter should be 2 cycles (Decrement request should come for 2 times before voltage decrement can happen)
- Choose path 7 of ring_osc adder
- Modulus (actual_perf - target_perf) = 100 count
- The external I2C slave can be programmed in 1 command
- Size of voltage value is 4 bits
- The voltage values are in MSB 4 bits, the LSB 4 bits have all '1'
- The I2C PMU has 10 bit addressing mode
- I2C Slave address is 32
- Address for Voltage select register for I2C is 4

The registers which needed to be programmed are described below:

Table 31. Control Register 1

Bit	Name	Value	Description
31:11	PMU_WAIT_CNT	64	Number of ref_clks to wait for PMU voltage change request to take effect
10	INTR_EN	1	0: disable (default), 1:Enable
9	NA	NA	
8:2	NUM_ITER	4	Number of iterations to adjust the voltage
1:0	MODE	2	0: disable(default), 1: Fixed Voltage adjust mode, 2: Continuous mode

Table 32. Control Register 2

Bit	Name	Value	Description
31:9	TIMER_CNT	Don't care	Wakeup Timer Counter initial value to perform periodic voltage tuning
8:2	ROSC_SA_CNT	10	The period in terms of number of ref_clks, during which performance counter is incremented
1:0	ROSC_START_DEL	2	Number of ref clocks to wait let ring oscillator settle

Table 33. Control Register 3

Bit	Name	Value	Description
31	SW_PROG_PMU_DONE	0	Status bit which driver code should write to let DVC know that PMU has been programmed. DVC will then clear this bit.
30	I2C_DONE_INTR_EN	0	Enable I2C interrupt to know when I2C transfer is done
29	PMU_VOLTAGE_PROG_INTR_EN	0	PMU voltage program ready interrupt enable
28	FIRMWARE_TGT_PERF_INTR_EN	1	Firmware trigger interrupt enable
27	NA		
26	I2C_HW_SW_PROG	0	'0' hardware programs PMU via I2C. '1' software code programs PMU via I2C
25	TIMER_EN	0	Enable Wakeup timer
24:22	HYST_CNTR	2	Number of decrement requests, after an increment request, to wait for, before voltage change applied
21	TRIG_PM_SA	0	Self clearing bit that if set causes one performance monitor sample to be taken
20:16	MUX_SEL	7	Select 1 of 32 path of ring _ oscillator adder
15	LONG_PATH_EN	0	0:not long path, 1:select long path for clock
14	RING_OSC_SEL	1	0:Speedo, 1: new ring oscillator

Bit	Name	Value	Description
13:11	NA	0	
10	NA	0	
9:0	VA_TH_H	100	(actual performance-target performance)>threshold, voltage tuning is done if enabled

Table 34. I2C Control Register

Bit	Name	Value	Description
17	CMD_1_2	1	1 or 2 Command for PMU
16:14	SIZE	4	Size of voltage values to match with PMU
13:11	SHIFT	2	Shift voltage values to match with PMU
10	ADDR_7BIT_10BIT	1	7 bit or 10 bit addressing
9:0	SLAVE_ID	32	External slave ID Address

Table 35. I2C Adder Data Register

Bit	Name	Value	Description
31:24	DATA2	Don't Care	Optional second data
23:16	ADDR2	Don't Care	Optional second address
15:8	DATA1	15	Default data
7:0	ADDR1	4	Address for voltage select

For driver code to control I2C register programming, Bit 26 & 31 of Control Register 3 I2C_HW_SW_PROG & SW_PROG_PMU_DONE should be 1. For Programming the I2C please read the I2C programming guidelines.

Steps involved in voltage change request for this use case are:

1. Software programs the above registers with the given values
2. Software triggers the voltage change request by programming Request register
3. DVC reads the LUT and programs the External I2C PMU with the safe voltage
4. DVC then interrupts the driver code by asserting FIRMWARE_TGT_PERF_INTR
5. Software gets the interrupt and clears it by writing '1' to the above bit.
6. DVC if enabled will be in the continuous mode adjusting voltage
7. Software needs to wait till DVC de-asserts BUSY signal in status register, to send more requests

9.2.6.2 Use Case 2

DVC gets a request from driver code for voltage change request and driver code controls the External PMU via I2C. The following are the requirements.

- DVC should be in Continuous adjustment mode
- PMU settling time is 64 reference clocks
- Number of iterations is 4

- Interrupt is enabled
- Wakeup timer is disabled
- Ring oscillator should be sampled for 10 ref clocks (ROSC_SA_CNT)
- Ring oscillator settling time (ROSC_STRAT_DEL) is 2 ref clocks
- Firmware should know when DVC has completed the change request
- Hysteresis counter should be 2 cycles (Decrement request should come for 2 times before voltage decrement can happen)
- Choose path 7 of ring_osc adder
- Modulus (actual performance - target performance) = 100 count
- The external I2C slave can be programmed in 1 command
- Size of voltage value is 4 bits
- The voltage values are in MSB 4 bits, the LSB 4 bits have all '1'
- The I2C PMU has 10 bit addressing mode
- I2C Slave address is 32
- Address for Voltage select register for I2C is 4

The registers which needed to be programmed are described below:

Table 36. Control Register 1

Bit	Name	Value	Description
31:11	PMU_WAIT_CNT	64	Number of ref_clks to wait for PMU voltage change request to take effect
10	INTR_EN	1	0: disable (default), 1:Enable
9	NA	NA	
8:2	NUM_ITER	4	Number of iterations to adjust the voltage
1:0	MODE	2	0: disable(default), 1: Fixed Voltage adjust mode, 2: Continuous mode

Table 37. Control Register 2

Bit	Name	Value	Description
31:9	TIMER_CNT	Don't care	Wakeup Timer Counter initial value to perform periodic voltage tuning
8:2	ROSC_SA_CNT	10	The period in terms of number of ref clocks, during which performance counter is incremented
1:0	ROSC_START_DEL	2	Number of ref clocks to wait let ring oscillator settle

Table 38. Control Register 3

Bit	Name	Value	Description
31	SW_PROG_PMU_DONE	0	Status bit which driver code should write to let DVC know that PMU has been programmed. DVC will then clear this bit.
30	I2C_DONE_INTR_EN	0	Enable I2C interrupt to know when I2C transfer is done

Bit	Name	Value	Description
29	PMU_VOLTAGE_PROG_INTR_EN	1	PMU voltage program ready interrupt enable
28	FIRMWARE_TGT_PERF_INTR_EN	1	Firmware trigger interrupt enable
27	NA		
26	I2C_HW_SW_PROG	1	'0' hardware programs PMU via I2C. '1' software programs PMU via I2C
25	TIMER_EN	0	Enable Wakeup timer
24:22	HYST_CNTR	2	Number of decrement requests, after an increment request, to wait for, before voltage change applied
21	TRIG_PM_SA	0	Self clearing bit that if set causes one performance monitor sample to be taken
20:16	MUX_SEL	7	Select 1 of 32 path of ring _ oscillator adder
15	LONG_PATH_EN	0	0:not long path, 1:select long path for clock
14	RING_OSC_SEL	1	0:Speedo, 1: new ring oscillator
13:11	NA	0	
10	NA	0	
9:0	VA_TH_H	100	(actual performance-target performance)>threshold, voltage tuning is done if enabled

Table 39. I2C Control Register

Bit	Name	Value	Description
17	CMD_1_2	1	1 or 2 Command for PMU
16:14	SIZE	4	Size of voltage values to match with PMU
13:11	SHIFT	2	Shift voltage values to match with PMU
10	ADDR_7BIT_10BIT	1	7 bit or 10 bit addressing
9:0	SLAVE_ID	32	External slave ID Address

Table 40. I2C Adder Data Register

Bit	Name	Value	Description
31:24	DATA2	Don't Care	Optional second data
23:16	ADDR2	Don't Care	Optional second address
15:8	DATA1	15	Default data
7:0	ADDR1	4	Address for voltage select

Steps involved in voltage change request for this use case are:

1. Software programs the above registers with the given values
2. Software triggers the voltage change request by programming Request register
3. DVC reads the LUT and asserts the interrupt bit PMU_VOLTAGE_PROG_INTR
4. Software then programs the external PMU via the required interface
5. Software then comes and writes '1' to SW_PROG_PMU_DONE bit in control register 3
6. This bit will be self-cleared by DVC and DVC will assert FIRMWARE_TGT_PERF_INTR
7. DVC if enabled will be in the continuous mode adjusting voltage
8. Whenever the voltage is ready to be sent to PMU, DVC asserts the PMU_VOLTAGE_PROG_INTR for driver code to handle the PMU
9. Software needs to wait till DVC de-asserts BUSY signal in status register, to send more requests

9.2.7 I2C Register Programming Sequence

Normal Mode

1. Register I2C_DIRECT_CNFG will be programmed 0x03 for 10 bit addressing or 0x02 for 7 bit addressing mode
2. Register I2C_DIRECT_CMD_ADDR0 will be programmed the slave Address. If External slave address is 0x49 driver code should program I2C CONTROL REG, SLAVE_ID field as 0x92 (left shift by 1). This is the way that the I2C master needs the slave address.
3. Register I2C_DIRECT_CMD_DATA1 will be programmed concatenated value of {DATA1, ADDR1} which is got from I2C ADDR DATA register bits15:8 & 7:0 respectively.
4. Register I2C_DIRECT_CNFG will be programmed 0x200 to trigger I2C transfer.
5. Register I2C_DIRECT_STATUS will be polled by the DVC to know the I2C transfer completion.

Packet Mode

PACKET_MODE_TRANSFER_EN of register DVC_I2C_CNFG_0 should be set to 1 while working in packet mode. Refer to the I2C section of this manual for complete details on the register programming sequence.

9.2.8 DVC Registers

9.2.8.1 DVC_CTRL_REG1_0

Control Register 1

This register maintains DVC configuration fields. There are three modes for DVC operations: disabled, fixed voltage adjust mode and continuous mode.

Disabled: In this mode DVC voltage adjustment operation is disabled.

Fixed Voltage Adjust Mode: In this mode the voltage adjustment is done only once.

Continuous Mode: In this mode the DVC updates the voltage continuously to track the PVT changes. PMU_WAIT_CNT will be used to wait for the PMU voltage to stabilize.

Offset: 000h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:11	0x0	PMU_WAIT_CNT: Number of ref_clks to wait for PMU voltage change request to take effect

Bit	Reset	Description
10	0x0	INTR_EN: Enable Interrupt 0: disable (default), 1:Enable 0 = DISABLE 1 = ENABLE
9	0x0	EXT_PMU: 0:not present, 1:present 0 = NOT_PRESENT 1 = PRESENT
8:2	0x0	NUM_ITER: Number of iterations to adjust the voltage
1:0	0x0	MODE: 0: disable(default), 1: Fixed Voltage adjust mode, 2: Continuous mode 0 = DISABLE 1 = FIX_MODE 2 = CONT_MODE

9.2.8.2 DVC_CTRL_REG2_0

Control Register 2

This register maintains counter values used to control the ring oscillator performance measurements.

Offset: 004h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:9	0x0	TIMER_CNT: Wakeup timer, in terms of number of ref clocks, for voltage adjustment process.
8:2	0x0	ROSC_SA_CNT: The period in terms of number of ref clocks, during which performance counter is incremented.
1:0	0x0	ROSC_START_DEL: Number of ref clocks to wait for the ring oscillator to settle.

9.2.8.3 DVC_CTRL_REG3_0

Control Register 3

This register is used to configure DVC parameters, like enabling events and choosing ring_osc.

TRIG_PM_SA will be used to take one sample of performance monitor. This will be used in characterization.

LONG_PATH_EN will choose a long feedback path or short feedback path for ring oscillator and will take into account the RC delay due to long path.

Offset: 008h | Read/Write: R/W | Reset: 0b0000x0000000000000xxxx0000000000

Bit	Reset	Description
31	0x0	SW_PROG_PMU_DONE: Status bit which driver code should write to let DVC know that PMU has been programmed. DVC will then clear this bit.
30	0x0	I2C_DONE_INTR_EN: Enable I2C interrupt when I2C transfer is done 0 = DISABLE 1 = ENABLE
29	0x0	PMU_VOLT_READY_INTR_EN: PMU voltage program ready interrupt enable 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
28	0x0	TGT_PERF_DONE_INTR_EN: Enable for target performance adjustment done interrupt 0 = DISABLE 1 = ENABLE
26	0x0	I2C_HW_SW_PROG: Select either hardware or driver code to program the PMU via I2C. 0 = HW 1 = SW
25	0x0	TIMER_EN: Enable Wakeup timer 0 = DISABLE 1 = ENABLE
24:22	0x0	HYST_CNTR: Number of decrement requests, after an increment request, to wait for, before voltage change applied
21	0x0	TRIG_PM_SA: Self clearing bit that if set causes one performance monitor sample to be taken
20:16	0x0	MUX_SEL: Select 1 of 32 path of ring _ oscillator adder
15	0x0	LONG_PATH_EN: 0:not long path, 1:select long path for clock 0 = DISABLE 1 = ENABLE
14	0x0	RING_OSC_SEL: Select between adder ring oscillator (0) and "speedo" ring oscillator (1). 0 = OLD 1 = NEW
9:0	0x0	VA_TH_H: (actual perf-target perf)>threshold, voltage tuning is done if enabled.

9.2.8.4 DVC_STATUS_REG_0

Status Register

This register gives the status of DVC operations.

Offset: 00ch | Read/Write: R/W | Reset: 0b00000000000011000000000000000000

Bit	Reset	Description
30	0x0	I2C_DONE_INTR: Interrupt to indicate I2C transfer is done
29	0x0	PMU_VOLT_READY_INTR: Interrupt indicating that voltage adjustment value is ready and can be programmed to PMU via I2C by driver code.
28	0x0	TGT_PERF_DONE_INTR: Interrupt to firmware to indicate voltage change has been completed
27	0x0	BUSY: DVC/PMU is busy adjusting voltage.
26	0x0	CARRY_OUT: Carry output from the adder of the new ring oscillator
25:22	0x0	I2C_STATUS: I2C Status bits
21	0x0	I2C_ERROR: Indicates error for I2C master in data transfer
20	0x0	TGT_PM_UNDERRUN: Measured performance count less than target performance count condition detected.
19	0x0	VADJ_ERR: Voltage adjustment exceeds the limit

Bit	Reset	Description
18:14	0x18	CURR_VOLT: Value of the voltage that has been applied.
13:0	0x0	PMON_VALUE: Performance monitor sample value for the last sample.

9.2.8.5 DVC_I2C_CTRL_REG_0

I2C Control Register

This is used to mask and shift the voltage select value that is being sent to PMU via I2C. Three sets of transactions can be sent.

Offset: 010h | Read/Write: R/W | Reset: 0b00010100010100010100000000000000

Bit	Reset	Description
31:29	0x0	MULTI_CMD: 1 or 2 or 3 Commands for writing to PMU-I2C slave. 000=> 1 command vsel1 only to core 001=> 2 commands vsel1 & vsel2 to the core & AO 010=> 3 commands vsel1 & vsel2 & vsel3 to the core, AO and CPU 011=> NA 100 => 2 commands vsel1 to the core, vsel2 is software controlled 101 & 110 = > NA 111 => 3 commands vsel1 to the core, vsel2 & vsel3 are software controlled
28:26	0x5	SIZE3: Size of vsel3 to match with PMU
25:23	0x0	SHIFT3: Shift vsel3 to match with PMU
22:20	0x5	SIZE2: Size of vsel2 to match with PMU
19:17	0x0	SHIFT2: Shift vsel2 to match with PMU
16:14	0x5	SIZE1: Size of vsel to match with PMU
13:11	0x0	SHIFT1: Shift vsel to match with PMU
10	0x0	ADDR_7BIT_10BIT: 7 bit or 10 bit addressing
9:0	0x0	SLAVE_ID: External slave ID Address

9.2.8.6 DVC_I2C_ADDR_DATA_REG_0

I2C Adder Data Register

This is used for sending the user defined address and data over I2C. It is used to address the appropriate I2C device (PMU) that will adjust the voltage to the core.

Offset: 014h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:24	0x0	DATA2: Optional second data
23:16	0x0	ADDR2: Optional second address
15:8	0x0	DATA1: Default data
7:0	0x0	ADDR1: Address for voltage select

9.2.8.7 DVC_REQ_REGISTER_0

Firmware ctrl Register 2

Firmware can use these bits to trigger DVC for a voltage change request.

Offset: 020h | Read/Write: R/W | Reset: 0b0000000

Bit	Reset	Description
6	0x0	REQ_VLD: Self clearing bit, which firmware can use to trigger DVC voltage change 0 = INVALID 1 = VALID
5:0	0x0	NORM_FREQ: firmware target performance

9.2.8.8 DVC_I2C_ADDR_DATA_REG_3_0

I2C Address Data 3 Register

Firmware can specify the address and data of the I2C device in this register.

Offset: 024h | Read/Write: R/W | Reset: 0b0000000000000000

Bit	Reset	Description
15:8	0x0	DATA3: Default Data
7:0	0x0	ADDR3: Address For Voltage select 3

9.2.8.9 DVC_I2C_CNFG_0

The master can now work in two modes.

- Normal mode (old)
- Packet mode (new)

In normal mode, the following registers need to be programmed:

- I2C_CNFG[9:0]
- I2C_CMD_ADDR0
- I2C_CMD_ADDR1
- I2C_CMD_DATA1
- I2C_CMD_DATA2
- I2C_STATUS

In packet-mode the following registers need to be programmed:

- I2C_CNFG[10]
- I2C_TX_PACKET_FIFO
- I2C_RX_PACKET_FIFO
- PACKET_TRANSFER_STATUS
- FIFO_CONTROL
- FIFO_STATUS
- INTERRUPT_MASK_REGISTER

- INTERRUPT_STATUS_REGISTER

Note: The FREQUENCY DIVISOR register (CLK_SOURCE_I2C register must be programmed as a function of the CLK_SOURCE selected for I2C as follows:

I2C_CLK = CLK_SOURCE.I2C / (8 * I2C FREQUENCY DIVISOR)

The I2C bus specification defines the minimum low period for the I2C_CLK as 4.7 s in standard mode and 1.3 s in fast mode. Because of this, the maximum I2C_CLK frequency in the standard mode can be 100 KHz but in fast mode, it is limited to 348 KHz, assuming I2C_CLK as rise and fall delays of 300 ns per the I2C specification.

The clock enable (bit-12 of CLK_OUT_ENB.L register) must also be given to I2C controller, before any of the registers are written.

IC Controller Configuration Register (Master)

I2C_CNFG register is used to configure the number of bytes to be transmitted or received, the slave device type either a 7-bit device or a 10-bit device, Enable mode to send Start-Byte or not, to select either a single slave transaction or two slave transactions, enable mode to handle devices that do not generate ACK.

Offset: 040h | Read/Write: R/W | Reset: 0b000000000000000

Bit	Reset	Description
14:12	0x0	DEBOUNCE_CNT: Debounce period for sda and scl lines 0 = No debounce 1 = 2T 2 = 4T 3 = 6T etc where T is the period of the fix PLL clk source coming to I2C.Maximum debounce period programmable is 14T.A debounce period of >50ns is desirable
11	0x0	NEW_MASTER_FSM: Write 1 to enable new master fsm 0 = old fsm 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. Values of other bits are not affected when this bit is 1, Cleared by hardware. Other bits of the register are masked for writes when this bit is programmed to one. Hence, firmware should first configure all other registers and bits [8:0] of I2C_CNFG register before the bit I2C_CNFG[9] is programmed to zero. 0 = NOP 1 = GO
8	0x0	NOACK: Enable mode to handle devices that do not generate ACK. 1 - do not look for an ack at the end of the Enable. 0 = DISABLE 1 = ENABLE
7	0x0	CMD2: Read/Write Command for Slave 2: 1 - Read Transaction; 0 - write Transaction. For a 7-bit slave address, this bit must match with the LSB of address byte for slave 2. Valid only when bit-4 of this register is set 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
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 slave2. 0 = DISABLE 1 = ENABLE
5	0x0	START: 1 = Yes, a Start byte needs to be sent. 0 = DISABLE 1 = ENABLE
4	0x0	SLV2: 1 - Enables a two slave transaction; 0 = No command for Slave 2 present 0 = DISABLE 1 = ENABLE
3:1	0x0	LENGTH: The Number of bytes to be transmitted per transaction 000= 1byte ... 111 = 8bytes; In a two slave transaction number of bytes should be programmed less than 011.
0	0x0	A_MOD: Address mode defines whether a 7-bit or a 10-bit slave address is programmed. 1 = 10-bit device address 0 = 7-bit device address 0 = SEVEN_BIT_DEVICE_ADDRESS 1 = TEN_BIT_DEVICE_ADDRESS

9.2.8.10 DVC_I2C_CMD_ADDR0_0

I2C Slave-1 Address

I2C_CMD_ADDR0 is programmed the 7 Bit or 10 Bit address of slave 1 with which the transaction is intended.

Offset: 044h | Read/Write: R/W | Reset: 0b0000000000

Bit	Reset	Description
9:0	0x0	ADDR0: In case of 7-Bit mode address is written in the I2C_CMD_ADDR0[7:1] and I2C_CMD_ADDR0[0] indicates the read/write transaction.I2C_CMD_ADDR0[0] bit must match with the I2C_CNFG[6]. In case of 10-Bit mode address is written in I2C_CMD_ADDR0[9:0] and I2C_CNFG[6] indicates the read/write transaction

9.2.8.11 DVC_I2C_CMD_ADDR1_0

I2C Slave-2 Address

I2C_CMD_ADDR1 is programmed the 7 Bit or 10 Bit address of slave 2 with which the transaction is intended.

Offset: 048h | Read/Write: R/W | Reset: 0b0000000000

Bit	Reset	Description
9:0	0x0	ADDR1: In case of 7-Bit mode address is written in the I2C_CMD_ADDR0[7:1] and I2C_CMD_ADDR0[0] indicates the read/write transaction.I2C_CMD_ADDR0[0] bit must match with the I2C_CNFG[7]. In case of 10-Bit mode address is written in I2C_CMD_ADDR0[9:0] and I2C_CNFG[7] indicates the read/write transaction.

9.2.8.12 DVC_I2C_CMD_DATA1_0

IC Controller Data 1: Transmit/Receive

The four Least Significant Bytes of Data to be transmitted is loaded into the register when I2C Master is in Write Mode.

The four Least Significant Bytes of Data are Read through this register when I2C Master is in Read mode.

Offset: 04ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:24	0x0	DATA4: Fourth data byte to be sent/received
23:16	0x0	DATA3: Third data byte to be sent/received
15:8	0x0	DATA2: Second data byte to be sent/received
7:0	0x0	DATA1: This register contains the first data byte to be sent/received.

9.2.8.13 DVC_I2C_CMD_DATA2_0

IC Controller Data 2: Transmit/Receive

The four Most Significant Bytes of Data to be transmitted is loaded into the register when I2C Master is in Write Mode.

The four Most Significant Bytes of Data are Read through this register when I2C Master is in Read mode.

Offset: 050h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:24	0x0	DATA8: Eighth data byte to be sent/received
23:16	0x0	DATA7: Seventh data byte to be sent/received
15:8	0x0	DATA6: Sixth data byte to be sent/received
7:0	0x0	DATA5: This register contains the Fifth data byte to be sent/received.

9.2.8.14 DVC_I2C_STATUS_0

IC Controller Status (Master)

I2C_STATUS gives the status of I2C master operation.

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

Bit	Reset	Description
8	0x0	BUSY: 1 = Busy. 0 = NOT_BUSY 1 = BUSY

Bit	Reset	Description
7:4	0x0	CMD2_STAT: Transaction for Slave2 for x byte failed. x is 'h0 to 'ha. All others invalid 0 = SL2_XFER_SUCCESSFUL 1 = SL2_NOACK_FOR_BYTE1 2 = SL2_NOACK_FOR_BYTE2 3 = SL2_NOACK_FOR_BYTE3 4 = SL2_NOACK_FOR_BYTE4 5 = SL2_NOACK_FOR_BYTE5 6 = SL2_NOACK_FOR_BYTE6 7 = SL2_NOACK_FOR_BYTE7 8 = SL2_NOACK_FOR_BYTE8 9 = SL2_NOACK_FOR_BYTE9 10 = SL2_NOACK_FOR_BYTE10
3:0	0x0	CMD1_STAT: Transaction for Slave1 for x byte failed. x is 'h0 to 'ha. All others invalid 0 = SL1_XFER_SUCCESSFUL 1 = SL1_NOACK_FOR_BYTE1 2 = SL1_NOACK_FOR_BYTE2 3 = SL1_NOACK_FOR_BYTE3 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

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. Normal mode registers and the operation are not affected with the packet mode interface changes. The transition from normal mode to packet mode is suggested because packet mode allows:

- No restriction on the number of bytes before and after the repeated start
- Number of bytes that can be transferred with a single cmd is not limited to 8 (though a packet can contain 4k bytes, because any number of packets can be pushed into the FIFO, there is no limit on the number of bytes that can be transferred)
- Transactions to different slaves can be chained together with repeat start and there is no limit on the no. of slaves it can address

9.2.8.15 DVC_I2C_TX_PACKET_FIFO_0

A packet contains header and payload. Header size is variable and could vary from 2 to 5 words. For I2C it is 3 words. The first two words of the header contain generic information. The third word contains I2C transaction-specific information. Payload contains actual data to be written to the slave. In case of read operation, payload is nil, hence the packet contains header only

Offset: 060h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	TX_PACKET: SW writes packets into this register A packet may contain generic header or I2C specific header or data

9.2.8.16 DVC_I2C_RX_FIFO_0

Header or I2C Specific Header or Data

Offset: 064h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	RD_DATA: SW Reads data from this register, causes pop

9.2.8.17 DVC_PACKET_TRANSFER_STATUS_0

Offset: 068h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
24	X	TRANSFER_COMPLETE: The packet transfer for which last packet is set has been completed 0 = UNSET 1 = SET
23:16	X	TRANSFER_PKT_ID: The current packet id for which the transaction is happening on the bus
15:4	X	TRANSFER_BYTENUM: The number of bytes transferred in the current packet
3	X	NOACK_FOR_ADDR: No ack recieved for the addr byte 0 = UNSET 1 = SET
2	X	NOACK_FOR_DATA: No ack recieved 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

9.2.8.18 DVC_FIFO_CONTROL_0

Offset: 06ch | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7:5	0x0	TX_FIFO_TRIG: Transmit FIFO trigger level 000 = 1 word, DMA trigger is asserted when at least one word empty in the FIFO 010 = 2 word, DMA trigger is asserted when at least 2 words empty in the FIFO
4:2	0x0	RX_FIFO_TRIG: Receive FIFO trigger level 000 = 1 word DMA trigger is asserted when at least one word full in the FIFO 010 = 2 word DMA trigger is asserted when at least 2 word full in the FIFO
1	0x0	TX_FIFO_FLUSH: 1= flush the TX FIFO, cleared after FIFO is flushed 0 = UNSET 1 = SET

Bit	Reset	Description
0	0x0	RX_FIFO_FLUSH: 1= flush the RX FIFO, cleared after FIFO is flushed 0 = UNSET 1 = SET

9.2.8.19 DVC_FIFO_STATUS_0

Offset: 070h | Read/Write: RO | Reset: 0bxxxxxxxx

Bit	Reset	Description
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

9.2.8.20 DVC_INTERRUPT_MASK_REGISTER_0

Offset: 074h | Read/Write: R/W | Reset: 0b0000000

Bit	Reset	Description
6	0x0	ALL_PACKETS_XFER_COMPLETE_INT_EN: 0 = DISABLE 1 = ENABLE
5	0x0	TFIFO_OVF_INT_EN: 0 = DISABLE 1 = ENABLE
4	0x0	RFIFO_UNF_INT_EN: 0 = DISABLE 1 = ENABLE
3	0x0	NOACK_INT_EN: 0 = DISABLE 1 = ENABLE
2	0x0	ARB_LOST_INT_EN: 0 = DISABLE 1 = ENABLE
1	0x0	TFIFO_DATA_REQ_INT_EN: 0 = DISABLE 1 = ENABLE
0	0x0	RFIFO_DATA_REQ_INT_EN: 0 = DISABLE 1 = ENABLE

9.2.8.21 DVC_INTERRUPT_STATUS_REGISTER_0

This register indicates the status bit for which the interrupt is set. If set, write 1 to clear it. However TFIFO_DATA_REQ, RFIFO_DATA_REQ fields depend on the FIFO trigger levels and cannot be cleared.

Offset: 078h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	PACKET_XFER_COMPLETE: A packet has been transferred successfully. TRANSFER_PKT_ID filed can be used to know the current byte under transfer. This bit can be masked by the IE field in the I2C specific header 0 = UNSET 1 = SET
6	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
1	0x0	TFIFO_DATA_REQ: TX FIFOdata req 0 = UNSET 1 = SET
0	0x0	RFIFO_DATA_REQ: RX FIFO data req 0 = UNSET 1 = SET

9.2.8.22 DVC_I2C_CLK_DIVISOR_REGISTER_0

The divisor value(N) must be programmed so that desired SCL freq = Clk source freq/12*N

Typically clk source frequency is fixed at 72 MHz.

Offset: 07ch | Read/Write: R/W | Reset: 0b0000000000000000

Bit	Reset	Description
15:0	0x0	I2C_CLK_DIVISOR_HSMODE: N= divide by n+1

9.2.8.23 DVC_VSEL_MAP_LUT_0

VSEL MAPPING Look Up Table

This buffer allows for mapping voltage select values. VSEL1 is determined corresponding to the performance of the core rail and based on that, VSEL2 and VSEL3 can be determined from this table.

This is an array of 32 identical register entries; the register fields below apply to each entry.

Offset: 080h..0ffh | Read/Write: R/W | Reset: 0b0000000000

Bit	Reset	Description
9:5	0x0	VSEL3: VSEL3 Corresponding to VSEL1
4:0	0x0	VSEL2: VSEL2 Corresponding to VSEL1

9.2.8.24 DVC_VLUT_0

Voltage Lookup Table

This is a LUT for voltage lookup. Software will pass an index which in turn gives a performance count, Vmin and Vmax. DVC will try to change the voltage to meet the performance count.

This is an array of 64 identical register entries; the register fields below apply to each entry.

Offset: 100h..1ffh | Read/Write: R/W | Reset: 0b00000000000000000000000000 | Default: 0000.0000

Bit	Reset	Description
23:10	0x0	PMCNT: Target performance count
9:5	0x0	VMIN: Minimum voltage selection value for a given frequency
4:0	0x0	VMAX: Maximum voltage selection value for a given frequency

10.0 AHB

10.1 AHB Bus

The AHB Bus conforms to the ARM® *AMBA*™ *Specification (Rev 2.0) Advanced High-performance Bus (AHB)* architecture as published by ARM.

AHB is a 32-bit multi-master bus. Despite the nomenclature, it is considered a second tier bus in the Tegra® 2 Processor, slower and less flexible than the AXI bus used by the CPU, or the memory client interface used by the high speed devices.

AHB clients in Tegra 2 include:

- The AHB DMA controller master
- The AHB memory controller slave
- The AVP cross-bar, both as a master and as a slave
- The APB DMA controller
- The USB-OTG controller
- The IDE controller
- The NAND flash controller
- The NOR/MIO controller
- Four SDMMC Controllers

10.2 AHB Bus Arbiter

The AHB Arbiter controls AHB bus master arbitration. This effectively forms a second level of arbitration for access to the memory controller through the AHB Slave Memory device, although AHB masters in some circumstances can use other AHB slaves, in particular access IRAM.

The controls presented here are largely for diagnostic purposes only. For suspect AHB performance problems, try disabling the other masters for the device with performance issues. In normal operation bus parking is usually enable, and all the masters are enabled.

Also see the AHB slave interface registers, where the AHB pre-fetch logic can be configured to enhance performance for devices doing sequential access.

10.2.1 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.

10.2.2 AHB Arbiter Registers

10.2.2.1 AHB_ARBITRATION_DISABLE_0

The AHB arbitration control register allows user to tweak arbitration behavior of the AHB arbiter.

- Disable 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: 000h | Read/Write: R/W | Reset: 0b0xxxxxxxxxx00000xx00000000000000

Bit	Reset	Description
31	0x0	DIS_BUS_PARK: 1 = disable bus parking. 0 = ENABLE 1 = DISABLE
20	0x0	SDMMC3: 1 = disable SDMMC3 from arbitration. 0 = ENABLE 1 = DISABLE
19	0x0	SDMMC2: 1 = disable SDMMC2 from arbitration. 0 = ENABLE 1 = DISABLE

Bit	Reset	Description
18	0x0	USB2: 1 = disable USB2 from arbitration. 0 = ENABLE 1 = DISABLE
17	0x0	USB3: 1 = disable USB3 from arbitration. 0 = ENABLE 1 = DISABLE
16	0x0	BSEA: 1 = disable BSEA from arbitration. 0 = ENABLE 1 = DISABLE
13	0x0	BSEV: 1 = disable BSEV from arbitration. 0 = ENABLE 1 = DISABLE
12	0x0	SDMMC4: 1 = disable SDMMC4 from arbitration. 0 = ENABLE 1 = DISABLE
11	0x0	SNOR: 1 = disable SNOR from arbitration. 0 = ENABLE 1 = DISABLE
10	0x0	NAND: 1 = disable NAND from arbitration. 0 = ENABLE 1 = DISABLE
9	0x0	SDMMC1: 1 = disable SDMMC1 from arbitration. 0 = ENABLE 1 = DISABLE
8	0x0	XIO: 1 = disable XIO 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	EIDE: 1 = disable EIDE 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

Bit	Reset	Description
1	0x0	COP: 1 = disable COP from arbitration. 0 = ENABLE 1 = DISABLE
0	0x0	CPU: 1 = disable CPU from arbitration. 0 = ENABLE 1 = DISABLE

10.2.2.2 AHB_ARBITRATION_PRIORITY_CTRL_0

The AHB arbiter implements a 2-level priority scheme. In the 1st level, arbitration is determined between the high and low priority group according to the priority weight; the higher the weight, the higher the winning rate of the high priority group.

In the 2nd level, within each of the high/low priority group, arbitration is determined in a round-robin fashion.

AHB Arbitration Priority Control Register

Offset: 004h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:29	0x0	AHB_PRIORITY_WEIGHT: AHB priority weight count. This 3-bit field is use to control the amount of attention (weight) given to the high priority group before switching to the low priority group.
28:0	0x0	AHB_PRIORITY_SELECT: 0 = low priority group.

10.2.2.3 AHB_ARBITRATION_USR_PROTECT_0

USR Protection Register

Offset: 008h | Read/Write: RO | Reset: 0bxxxxxxxxxx

Bit	Reset	Description
8	X	CACHE: Abort on USR mode access to Cache memory space 0 = ABT_DIS 1 = ABT_EN
7	X	ROM: Abort on USR mode access to internal ROM memory space 0 = ABT_DIS 1 = ABT_EN
6	X	APB: Abort on USR mode access to APB memory space 0 = ABT_DIS 1 = ABT_EN
5	X	AHB: Abort on USR mode access to AHB memory space 0 = ABT_DIS 1 = ABT_EN
4	X	PPSB: Abort on USR mode access to PPSB memory space 0 = ABT_DIS 1 = ABT_EN
3	X	IRAMD: Abort on USR mode access to iRAMd memory space 0 = ABT_DIS 1 = ABT_EN
2	X	IRAMC: Abort on USR mode access to iRAMc memory space 0 = ABT_DIS

Bit	Reset	Description
		1 = ABT_EN
1	X	IRAMB: Abort on USR mode access to iRAMb memory space 0 = ABT_DIS 1 = ABT_EN
0	X	IRAMA: Abort on USR mode access to iRAMa memory space 0 = ABT_DIS 1 = ABT_EN

10.3 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.

10.3.1 AHB Gizmo Registers

10.3.1.1 AHB_GIZMO_AHB_MEM_0

AHB Master/Slave Gizmo Register

Given below is a description of each configuration field, what they are used for and the pros and cons of using them.

> **Note:** If the gizmo configuration bits are changed while the particular gizmo is transferring data, behavior of the system is non-deterministic.

AHB gizmo configuration bit definition:

(1) AHB Master Gizmo

1. MAX_AHB_BURSTSIZE: Controls the maximum burst size that this gizmo can generate on the AHB bus. For the current system, this field should be set to burst-of-8.
2. IMMEDIATE: Controls how quickly an AHB write request on the bus can start.
   - If 1, gizmo will start an AHB write request once the IP side provides one beat of data of a burst. For IP that can provide quick continuous burst data, this setting provides parallelism between AHB request and IP data transfer. However, for IP that cannot provide quick burst data, this setting will force this gizmo to hold the AHB bus longer, reducing bus efficiency.
   - If 0, gizmo will wait until all beats of data of a burst are provided before starting an AHB write request. With this setting, AHB bus efficiency is realized but IP latency and efficiency may be reduced.
3. RD_DATA: Controls how read data will be returned from the gizmo to the IP side.
   - If 1, all betas of data of a burst have to be available before it indicates to the IP logic that read data is ready. This setting ensures there's no wait-state (bubble) between read data burst, but it will increase latency.
   - If 0, each beat of data of a burst will be sent from gizmo to the IP logic immediately. This setting will reduce latency, but can create non-consecutive data burst or bubble between data.
4. REQ_NEG_CNT: Provides a way to limit (in terms of number of AHB bus clock) how fast/often this master can request the AHB bus. The bigger the number, the slower the rate of AHB request.

(2) AHB Slave Gizmo

1. ENABLE_SPLIT: Controls enabling the split feature of the AHB bus.
   - If 1, gizmo will always generate split response for a read. If ' DON’T_SPLIT_AHB_WR' is set to 0, gizmo also generates split response if it cannot accept write request from the AHB master anymore. This setting can increase AHB bus utilization since it allows other AHB masters to talk to other AHB slaves while this original AHB slave is fetching read data. However, the flip side is that it takes longer time for the original AHB master to get

the read data, because the original AHB master has to re-arbitrate for the AHB bus again in order to get to the data it asked for.

- If 0, gizmo will not generate split response. Instead, it will hold on to the AHB bus until all the requested read data is returned back to the AHB master. This setting will reduce read latency to the master, but decrease AHB bus utilization.

2. FORCE_TO_AHB_SINGLE: Controls how the gizmo treats the AHB master's burst request.
   - If 1, gizmo will break up the burst request internally into individual single word requests.
   - If 0, gizmo will burst request internally as burst request.
3. ENB_FAST_REARBITRATE: Controls when the gizmo can allow the original read requested AHB master to re-arbitrate for the AHB bus again so the AHB master can retrieve the originally requested read data.
   - If 1, once first read data of a burst is in the slave gizmo's FIFO, it will allow the original AHB master to re-arbitrate, thus, allowing arbitration to happen in parallel with subsequent read data of a burst. However, if the data burst has wait-states or bubbles, then this setting will decrease AHB bus utilization because the slave gizmo will hold on to the AHB bus longer.
   - If 0, all read data of a burst must be in the slave gizmo's FIFO before it allows the original AHB master to re-arbitrate to retrieve the read data. This increases read data latency, and also increase AHB bus utilization.
4. IP_WR_REQ_IMMEDIATE: Controls when gizmo will start write data request to the IP logic.
   - If 1, gizmo will start write request to the IP logic once it gets one write data of a burst from the AHB side. This setting will create non-consecutive/bubbles in a write data burst.
   - If 0, gizmo will start write request to the IP logic only when it gets all write data of a burst from the AHB side. This setting will create consecutive data burst (no bubbles or wait-states).
5. MAX_IP_BURSTSIZE: Controls the maximum burst size that this gizmo can generate to the IP logic.
6. ACCEPT_AHB_WR_ALWAYS: Controls how the slave gizmo will treat AHB write request.
   - If 1, gizmo will always accept a write request without checking whether it's FIFOs can accept the write or not. This setting can reduce bus utilization, but can reduce the rate of AHB retry.
   - If 0, gizmo will check its FIFOs to make sure they have room before accepting the AHB write request. This setting increase bus utilization, but can create a lot of AHB retry.
7. DON'T_SPLIT_AHB_WR: Controls whether to split AHB write request when the slave gizmo determines it cannot accept the write request.
   - If 1 and when ENABLE_SPLIT=1, gizmo will generate split for write if its FIFOs are not ready to accept the AHB write request or data. This setting can improve AHB bus utilization as there are no continuous AHB master retries on the bus.
   - If 0, gizmo will generate retry response for write if its FIFOs are not ready to accept the AHB write request. Software should leave this bit at 0 since this feature has not been proven.

AHB Gizmo AHB-DMA/Memory Control Register

Offset: 00ch | Read/Write: R/W | Reset: 0b00000000xxxxx010xxxxxxxx11xxx001

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clk count between requests from this AHB master.
18	0x0	IMMEDIATE: AHB master gizmo (AHB-DMA) - Start AHB write request immediately 1 = start the AHB write request immediately as soon as the device has put one write data in the AHB gizmos queue. 0 = start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	0x2	MAX_AHB_BURSTSIZE: AHB master gizmo (AHB-DMA) - Maximum allowed AHB burst size. 00 = single transfer. 01 = burst-of-4. 10 = burst-of-8 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS
7	0x1	DON'T_SPLIT_AHB_WR: AHB slave gizmo (memory controller) - don't split AHB write transaction 1 = don't split AHB write transaction ever. 0 (and enable_split=1) = allow AHB write transaction to be split. 0 = ENABLE 1 = DISABLE
6	0x1	ACCEPT_AHB_WR_ALWAYS: AHB slave gizmo (memory controller) - Accept AHB write request always. 1= always accept AHB write request without checking whether there is room in the queue to store the write data. Bypass Memory Controller AHB slave gizmo write queue. 0 = accept AHB write request only when there's enough room in the queue to store all the write data. Memory controller AHB slave gizmos write queue is used in this case. 0 = ACCEPT_ON_CHECK 1 = ACCEPT_ON_NOCHECK
2	0x0	ENB_FAST_REARBITRATE: AHB slave gizmo - Enable fast re-arbitration. 1 = allow AHB master re-arbitration as soon as the device returns one read data into the gizmos queue. 0 = allow AHB master re-arbitration only when the device returns all read data into the gizmos queue. 0 = DISABLE 1 = ENABLE
1	0x0	FORCE_TO_AHB_SINGLE: AHB slave gizmo (memory controller) - Force all AHB transaction to single data request transaction 1 = force to single data transaction always. 0 = don't force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: /AHB slave gizmo (memory controller) - Enable splitting AHB transaction. 1 = enable 0 = disable. 0 = DISABLE 1 = ENABLE

10.3.1.2 AHB_GIZMO_APB_DMA_0

AHB Gizmo APB-DMA Control Register

Offset: 010h | Read/Write: R/W | Reset: 0b00000000xxxx1010

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clk count between requests from this AHB master.
19	0x1	RD_DATA: AHB master gizmo - Pack all AHB read data. 1 = wait for all requested read data to be in the AHB gizmos queue before returning the data back to the IP. 0 = transfer each read data from the AHB to the IP immediately. 0 = NO_WAIT 1 = WAIT
18	0x0	IMMEDIATE: AHB master gizmo (AHB-DMA) - Start AHB write request immediately. 1 = start the AHB write request immediately as soon as the device has put one write data in the AHB gizmos queue. 0 = start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	0x2	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer. 01 = burst-of-4. 10 = burst-of-8. 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS

10.3.1.3 AHB_GIZMO_IDE_0

AHB Gizmo IDE Control Register

Offset: 018h | Read/Write: R/W | Reset: 0b00000000xxxx0010xxxxxxxx10111111

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter use to indicate the minimum number of clk count between requests from this AHB master.
19	0x0	RD_DATA: AHB master gizmo - Pack all AHB read data. 1 = wait for all requested read data to be in the AHB gizmos queue before returning the data back to the IP. 0 = transfer each read data from the AHB to the IP immediately. 0 = NO_WAIT 1 = WAIT
18	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

Bit	Reset	Description
7	0x1	DON'T_SPLIT_AHB_WR: AHB slave gizmo - don't split AHB write transaction. 1 = don't split AHB write transaction ever. 0 (and enable_split=1) = allow AHB write transaction to be split. 0 = ENABLE 1 = DISABLE
6	0x0	ACCEPT_AHB_WR_ALWAYS: AHB slave gizmo - Accept AHB write request always. 1 = always accept AHB write request without checking whether there is room in the queue to store the write data. 0 = accept AHB write request only when there's enough room in the queue to store all the write data. 0 = ACCEPT_ON_CHECK 1 = ACCEPT_ON_NOCHECK
5:4	0x3	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	IP_WR_REQ_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.
2	0x1	ENB_FAST_REARBITRATE: AHB slave gizmo - Enable fast re-arbitration. 1 = allow AHB master re-arbitration as soon as the device returns one read data into the gizmos queue.0 = allow AHB master re-arbitration only when the device returns all read data into the gizmos queue. 0 = DISABLE 1 = ENABLE
1	0x1	FORCE_TO_AHB_SINGLE: AHB slave gizmo - Force all AHB transaction to single data request transaction. 1 = force to single data transaction always. 0 = don't force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 1 = enable, 0 = disable. 0 = DISABLE 1 = ENABLE

10.3.1.4 AHB_GIZMO_USB_0

AHB Gizmo USB Control Register

Offset: 01ch | Read/Write: R/W | Reset: 0b00000000xxxx0010xxxxxxxx10xx0011

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clk count between requests from this AHB master.
19	0x0	RD_DATA: AHB master gizmo - Pack all AHB read data. 1 = wait for all requested read data to be in the AHB gizmos queue before returning the data back to the IP. 0 = transfer each read data from the AHB to the IP immediately. 0 = NO_WAIT 1 = WAIT

Bit	Reset	Description
18	0x0	IMMEDIATE: AHB master gizmo (AHB-DMA) - Start AHB write request immediately. 1 = start the AHB write request immediately as soon as the device has put one write data in the AHB gizmos queue. 0 = start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	0x2	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer. 01 = burst-of-4. 10 = burst-of-8. 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS
7	0x1	DON'T_SPLIT_AHB_WR: AHB slave gizmo - don't split AHB write transaction. 1 = don't split AHB write transaction ever. 0 (and enable_split=1) = allow AHB write transaction to be split. 0 = ENABLE 1 = DISABLE
6	0x0	ACCEPT_AHB_WR_ALWAYS: AHB slave gizmo - Accept AHB write request always. 1 = always accept AHB write request without checking whether there is room in the queue to store the write data. 0 = accept AHB write request only when there's enough room in the queue to store all the write data. 0 = ACCEPT_ON_CHECK 1 = ACCEPT_ON_NOCHECK
3	0x0	IP_WR_REQ_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	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	0x1	FORCE_TO_AHB_SINGLE: AHB slave gizmo - Force all AHB transaction to single data request transaction. 1 = force to single data transaction always. 0 = don't force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 1 = enable 0 = disable 0 = DISABLE 1 = ENABLE

10.3.1.5 AHB_GIZMO_AHB_XBAR_BRIDGE_0

AHB Gizmo AHB XBAR Bridge Control Register

Offset: 020h | Read/Write: R/W | Reset: 0b10001101

Bit	Reset	Description
7	0x1	DON'T_SPLIT_AHB_WR: AHB slave gizmo - don't split AHB write transaction. 1 = don't split AHB write transaction ever. 0 (and enable_split=1) = allow AHB write transaction to be split. 0 = ENABLE 1 = DISABLE
6	0x0	ACCEPT_AHB_WR_ALWAYS: AHB slave gizmo - Accept AHB write request always. 1 = always accept AHB write request without checking whether there is room in the queue to store the write data. 0 = accept AHB write request only when there's enough room in the queue to store all the write data. 0 = ACCEPT_ON_CHECK 1 = ACCEPT_ON_NOCHECK
5:4	0x0	MAX_IP_BURSTSIZE: AHB slave gizmo - Maximum allowed IP burst size. 00 = single transfer. 01 = burst-of-4. 10 = burst-of-8. 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS
3	0x1	IMMEDIATE: AHB slave gizmo - Start write request to device immediately. 1 = start write request on the device side as soon as the AHB master puts data into the gizmos queue. 0 = start the device write request only when the AHB master has placed all write data into the gizmos queue. 0 = DISABLE 1 = ENABLE
2	0x1	ENB_FAST_REARBITRATE: AHB slave gizmo - Enable fast re-arbitration. 1 = allow AHB master re-arbitration as soon as the device returns one read data into the gizmos queue. 0 = allow AHB master re-arbitration only when the device returns all read data into the gizmos queue. 0 = DISABLE 1 = ENABLE
1	0x0	FORCE_TO_AHB_SINGLE: AHB slave gizmo - Force all AHB transaction to single data request transaction. 1 = force to single data transaction always. 0 = don't force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 1 = enable 0 = disable 0 = DISABLE 1 = ENABLE

10.3.1.6 AHB_GIZMO_CPU_AHB_BRIDGE_0

AHB Gizmo CPU AHB Bridge Control Register

Offset: 024h | Read/Write: R/W | Reset: 0b00000000xxxx0110

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter use to indicate the minimum number of clk count between requests from this AHB master
19	0x0	RD_DATA: AHB master gizmo - Pack all AHB read data. 1 = wait for all requested read data to be in the AHB gizmos queue before returning the data back to the IP. 0 = transfer each read data from the AHB to the IP immediately. 0 = NO_WAIT 1 = WAIT
18	0x1	IMMEDIATE: AHB master gizmo (AHB-DMA) - Start AHB write request immediately. 1 = start the AHB write request immediately as soon as the device has put one write data in the AHB gizmos queue. 0 = start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	0x2	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer. 01 = burst-of-4. 10 = burst-of-8. 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS

10.3.1.7 AHB_GIZMO_COP_AHB_BRIDGE_0

AHB Gizmo COP AHB Bridge Control Register

Offset: 028h | Read/Write: R/W | Reset: 0b00000000xxxx0110

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter use to indicate the minimum number of clk count between requests from this AHB master
19	0x0	RD_DATA: AHB master gizmo - Pack all AHB read data. 1 = wait for all requested read data to be in the AHB gizmos queue before returning the data back to the IP. 0 = transfer each read data from the AHB to the IP immediately. 0 = NO_WAIT 1 = WAIT
18	0x1	IMMEDIATE: AHB master gizmo (AHB-DMA) - Start AHB write request immediately. 1 = start the AHB write request immediately as soon as the device has put one write data in the AHB gizmos queue. 0 = start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	0x2	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer. 01 = burst-of-4. 10 = burst-of-8. 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS

10.3.1.8 AHB_GIZMO_XBAR_APB_CTLR_0

AHB Gizmo XBAR APB Control Register

Offset: 02ch | Read/Write: R/W | Reset: 0b001

Bit	Reset	Description
5:4	0x0	MAX_IP_BURSTSIZE: AHB slave gizmo - Maximum allowed IP burst size. 00 = single transfer. 01 = burst-of-4. 10 = burst-of-8. 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS
3	0x1	IMMEDIATE: AHB slave gizmo - Start write request to device immediately. 1 = start write request on the device side as soon as the AHB master puts data into the gizmos queue. 0 = start the device write request only when the AHB master has placed all write data into the gizmos queue. 0 = DISABLE 1 = ENABLE

10.3.1.9 AHB_GIZMO_VCP_AHB_BRIDGE_0

AHB Gizmo VCP AHB Bridge Control Register

Offset: 030h | Read/Write: R/W | Reset: 0b00000000xxxx0110 |

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter use to indicate the minimum number of clk count between requests from this AHB master.
19	0x0	RD_DATA: AHB master gizmo - Pack all AHB read data. 1 = wait for all requested read data to be in the AHB gizmos queue before returning the data back to the IP. 0 = transfer each read data from the AHB to the IP immediately. 0 = NO_WAIT 1 = WAIT
18	0x1	IMMEDIATE: AHB master gizmo (AHB-DMA) - Start AHB write request immediately. 1 = start the AHB write request immediately as soon as the device has put one write data in the AHB gizmos queue. 0 = start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	0x2	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer. 01 = burst-of-4. 10 = burst-of-8. 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS

10.3.1.10 AHB_GIZMO_NAND_0

AHB Gizmo NAND Control Register

Offset: 03ch | Read/Write: R/W | Reset: 0b00000000xxxx1010

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter use to indicate the minimum number of clk count between requests from this AHB master.

Bit	Reset	Description
19	0x1	RD_DATA: AHB master gizmo - Pack all AHB read data. 1 = wait for all requested read data to be in the AHB gizmos queue before returning the data back to the IP. 0 = transfer each read data from the AHB to the IP immediately. 0 = NO_WAIT 1 = WAIT
18	0x0	IMMEDIATE: AHB master gizmo (AHB-DMA) - Start AHB write request immediately. 1 = start the AHB write request immediately as soon as the device has put one write data in the AHB gizmos queue. 0 = start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	0x2	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer. 01 = burst-of-4. 10 = burst-of-8. 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS

10.3.1.11 AHB_GIZMO_SDMMC4_0

This is now SDMMC4 master gizmo configuration

AHB Gizmo SDMMC4 Control Register

Offset: 044h | Read/Write: R/W | Reset: 0b00000000xxxxx010xxxxxxxx10xxx111

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter use to indicate the minimum number of clk count between requests from this AHB master.
18	0x0	IMMEDIATE: AHB master gizmo - Start AHB write request immediately. 1 = start the AHB write request immediately as soon as the device puts data in the AHB gizmos queue. 0 = start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	0x2	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer. 01 = burst-of-4. 10 = burst-of-8. 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS
7	0x1	DONT_SPLIT_AHB_WR: AHB slave gizmo - Dont split AHB write transaction. 1 = dont split AHB write transaction ever. 0 (and enable_split=1) = allow AHB write transaction to be split. 0 = ENABLE 1 = DISABLE
6	0x0	ACCEPT_AHB_WR_ALWAYS: AHB slave gizmo - Accept AHB write request always. 1 = always accept AHB write request without checking whether there is room in the queue to store the write data. 0 = accept AHB write request only when there's enough room in the queue to store all the write data. 0 = ACCEPT_ON_CHECK 1 = ACCEPT_ON_NOCHECK
2	0x1	ENB_FAST_REARBITRATE: AHB slave gizmo - Enable fast re-arbitration. 1 = allow AHB master re-arbitration as soon as the device returns one read data into the gizmos queue. 0 = allow AHB master re-arbitration only when the device returns all read data into the gizmos queue. 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
1	0x1	FORCE_TO_AHB_SINGLE: AHB slave gizmo - Force all AHB transaction to single data request transaction. 1 = force to single data transaction always. 0 = don't force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 1 = enable 0 = disable 0 = DISABLE 1 = ENABLE

10.3.1.12 AHB_GIZMO_XIO_0

AHB Gizmo XIO Control Register

Offset: 048h | Read/Write: R/W | Reset: 0b00000000xxxx0100

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter use to indicate the minimum number of clk count between requests from this AHB master.
19	0x0	RD_DATA: AHB master gizmo - Pack all AHB read data. 1 = wait for all requested read data to be in the AHB gizmos queue before returning the data back to the IP. 0 = transfer each read data from the AHB to the IP immediately. 0 = NO_WAIT 1 = WAIT
18	0x1	IMMEDIATE: AHB master gizmo (AHB-DMA) - Start AHB write request immediately. 1 = start the AHB write request immediately as soon as the device has put one write data in the AHB gizmos queue. 0 = start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	0x0	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

10.3.1.13 AHB_GIZMO_BSEV_0

This is now VDE's BSEV master gizmo configuration.

AHB Gizmo BSE Control Register

Offset: 060h | Read/Write: R/W | Reset: 0b00000000xxxx0010

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter use to indicate the minimum number of clk count between requests from this AHB master.
19	0x0	RD_DATA: AHB master gizmo - Pack all AHB read data. 1 = wait for all requested read data to be in the AHB gizmos queue before returning the data back to the IP. 0 = transfer each read data from the AHB to the IP immediately. 0 = NO_WAIT 1 = WAIT

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

10.3.1.14 AHB_GIZMO_BSEA_0

VDE's BSEA master gizmo configuration

AHB Gizmo BSEA Control Register

Offset: 070h | Read/Write: R/W | Reset: 0b00000000xxxxx010

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter use to indicate the minimum number of clk count between requests from this AHB master.
18	0x0	IMMEDIATE: AHB master gizmo - Start AHB write request immediately. 1 = start the AHB write request immediately as soon as the device puts data in the AHB gizmos queue. 0 = start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. NOTE: THIS SHOULD NEVER BE SET TO ENABLE (BSEV requires this bit to be 0) 0 = DISABLE 1 = ENABLE
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

10.3.1.15 AHB_GIZMO_NOR_0

AHB Gizmo AHB NOR flash Control Register

Offset: 074h | Read/Write: R/W | Reset: 0b10xxx101

Bit	Reset	Description
7	0x1	DON'T_SPLIT_AHB_WR: AHB slave gizmo - don't split AHB write transaction. 1 = don't split AHB write transaction ever. 0 (and enable_split=1) = allow AHB write transaction to be split. 0 = ENABLE 1 = DISABLE

Bit	Reset	Description
6	0x0	ACCEPT_AHB_WR_ALWAYS: AHB slave gizmo - Accept AHB write request always. 1 = always accept AHB write request without checking whether there is room in the queue to store the write data. 0 = accept AHB write request only when there's enough room in the queue to store all the write data. 0 = ACCEPT_ON_CHECK 1 = ACCEPT_ON_NOCHECK
2	0x1	ENB_FAST_REARBITRATE: AHB slave gizmo - Enable fast re-arbitration. 1 = allow AHB master re-arbitration as soon as the device returns one read data into the gizmos queue. 0 = allow AHB master re-arbitration only when the device returns all read data into the gizmos queue. 0 = DISABLE 1 = ENABLE
1	0x0	FORCE_TO_AHB_SINGLE: AHB slave gizmo - Force all AHB transaction to single data request transaction. 1 = force to single data transaction always. 0 = don't force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 1 = enable 0 = disable 0 = DISABLE 1 = ENABLE

10.3.1.16 AHB_GIZMO_USB2_0

This is now USB2 master gizmo configuration

AHB Gizmo USB2 Control Register

Offset: 078h | Read/Write: R/W | Reset: 0b00000000xxxxx010xxxxxxxx10xxx111

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter use to indicate the minimum number of clk count between requests from this AHB master.
18	0x0	IMMEDIATE: AHB master gizmo - Start AHB write request immediately. 1 = start the AHB write request immediately as soon as the device puts data in the AHB gizmos queue. 0 = start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	0x2	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer. 01 = burst-of-4. 10 = burst-of-8. 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS
7	0x1	DONT_SPLIT_AHB_WR: AHB slave gizmo - Dont split AHB write transaction. 1 = dont 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 theres 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 = dont force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 1 = enable 0 = disable 0 = DISABLE 1 = ENABLE

10.3.1.17 AHB_GIZMO_USB3_0

USB3 master gizmo configuration

AHB Gizmo USB3 Control Register

Offset: 07ch | Read/Write: R/W | Reset: 0b00000000xxxxx010xxxxxxxx10xxx111

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter use to indicate the minimum number of clk count between requests from this AHB master.
18	0x0	IMMEDIATE: AHB master gizmo - Start AHB write request immediately. 1 = start the AHB write request immediately as soon as the device puts data in the AHB gizmos queue. 0 = start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	0x2	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer. 01 = burst-of-4. 10 = burst-of-8. 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS
7	0x1	DONT_SPLIT_AHB_WR: AHB slave gizmo – Don't split AHB write transaction. 1 = don't split AHB write transaction ever. 0 (and enable_split=1) = allow AHB write transaction to be split. 0 = ENABLE 1 = DISABLE
6	0x0	ACCEPT_AHB_WR_ALWAYS: AHB slave gizmo - Accept AHB write request always. 1 = always accept AHB write request without checking whether there is room in the queue to store the write data. 0 = accept AHB write request only when theres 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

Bit	Reset	Description
		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 = dont force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 1 = enable 0 = disable 0 = DISABLE 1 = ENABLE

10.3.1.18 AHB_GIZMO_SDMMC1_0

SDMMC1 master gizmo configuration

AHB Gizmo SDMMC1 Control Register

Offset: 080h | Read/Write: R/W | Reset: 0b00000000xxxxx010xxxxxxxx10xxx111

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter use to indicate the minimum number of clk count between requests from this AHB master.
18	0x0	IMMEDIATE: AHB master gizmo - Start AHB write request immediately. 1 = start the AHB write request immediately as soon as the device puts data in the AHB gizmos queue. 0 = start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
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 - Dont split AHB write transaction. 1 = dont 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 theres 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 = dont force to single data transaction. 0 = NOT_SINGLE_DATA

Bit	Reset	Description
		1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 1 = enable 0 = disable 0 = DISABLE 1 = ENABLE

10.3.1.19 AHB_GIZMO_SDMMC2_0

SDMMC2 master gizmo configuration

AHB Gizmo SDMMC2 Control Register

Offset: 084h | Read/Write: R/W | Reset: 0b00000000xxxxx010xxxxxxxx10xxx111

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter use to indicate the minimum number of clk count between requests from this AHB master.
18	0x0	IMMEDIATE: AHB master gizmo - Start AHB write request immediately. 1 = start the AHB write request immediately as soon as the device puts data in the AHB gizmos queue. 0 = start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE; 1 = ENABLE
17:16	0x2	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer. 01 = burst-of-4. 10 = burst-of-8. 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS
7	0x1	DONT_SPLIT_AHB_WR: AHB slave gizmo - Dont split AHB write transaction. 1 = dont 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 theres 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 = dont force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 1 = enable 0 = disable 0 = DISABLE 1 = ENABLE

10.3.1.20 AHB_GIZMO_SDMMC3_0

SDMMC3 master gizmo configuration

AHB Gizmo SDMMC3 Control Register

Offset: 088h | Read/Write: R/W | Reset: 0b00000000xxxxx010xxxxxxxx10xxx111

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter use to indicate the minimum number of clk count between requests from this AHB master.
18	0x0	IMMEDIATE: AHB master gizmo - Start AHB write request immediately. 1 = start the AHB write request immediately as soon as the device puts data in the AHB gizmos queue. 0 = start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	0x2	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer. 01 = burst-of-4. 10 = burst-of-8. 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS
7	0x1	DONT_SPLIT_AHB_WR: AHB slave gizmo - Dont split AHB write transaction. 1 = dont 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 theres 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 = dont force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 1 = enable 0 = disable 0 = DISABLE; 1 = ENABLE

10.4 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.

10.4.1 AHB Memory Pre-Fetcher

The AHB memory controller slave contains a pre-fetcher block. This is intended to improve performance for AHB masters performing sequential reads from DRAM. Performance can be a problem because the AHB bus protocol only allows a single outstanding read request from a master, so the round trip latency limits bandwidth. As there are two level of arbitration, firstly on AHB and secondly within the memory controller, this latency can become large on a busy system.

There are four such pre-fetchers, allowing this function to be active for up to four AHB masters.

When the pre-fetcher is enabled, the first read request initiates a speculative read of up to 128 bytes, and subsequent linear reads will return the data directly to the AHB master without going through the memory controller to DRAM.

10.4.1.1 Pre-Fetch Invalidate

The pre-fetch buffer's contents are invalidated under any of the following conditions, in all cases a read refers to a DRAM read by the assigned HB master:

- The programmed time-out value is reached since the last read
- A read occurs which is not sequential with the previous read
- A read occurs which is not the same size as the previous read, unless the corresponding DISABLE_CHECK_SIZE_MASTERx bit is set.
- A read occurs which is past the end of the pre-fetch buffer (this will trigger a new fill of the buffer).
- The pre-fetcher is disabled. A momentary disable and then re-enable can be used to invalidate the buffer.

10.4.1.2 Pre-Fetch Buffer Coherency

As the pre-fetch buffer contains a copy of data in DRAM, there is an inherent coherency risk if the DRAM data is updated. The hardware invalidation mechanisms provide some protection here, but there remains risk of subtle problems arising from this hardware. If an AHB master shows errors that appear to arise from stale data then a reasonable experiment is to disable the pre-fetcher to see if that cures the problem. If so, the following guidelines may help.

Problems have been seen for control structures such as USB Transfer Descriptors.

Two approaches can resolve this problem:

1. Pad the data

As the pre-fetcher invalidates the buffer for any non-sequential read, placing some padding will prevent pre-fetch issues. A problem can be triggered by using something like this:

```
struct dtd { unsigned HW_descriptor[K] ; } ; struct dtd dtd_array[N] ;
```

This makes the HW descriptors structures as read by the USB DMA sequential, and so pre-fetchable.
If the definition is modified as:

```
struct dtd { unsigned HW_descriptor[K] ; unsigned Padding ; } ; struct dtd dtd_array[N] ;
```

Then reads by USB DMA of consecutive HW descriptors become **not** sequential and so there is no coherency issue as the pre-fetch buffer will be invalidated.

2. Invalidate the pre-fetch buffer

After updating a memory structure, the pre-fetch buffer can be invalidated by disabling the pre-fetcher and then immediately re-enabling it.

10.4.2 AHB Memory Controller Slave Registers

10.4.2.1 AHB_AHB_MEM_PREFETCH_CFG_X_0

If DISABLE_CHECK_SIZE is 0, then only read requests that have the exact same size as the original read request that kick-started the prefetch process will cause a "hit". In addition, the address must be the exact next one in the sequence.

For instance, if the first request on a prefetch-enabled AHB Master arrives with SIZE=ONE_BYTE and ADDR[31:0]=0x3, then the next access must have SIZE=ONE_BYTE and ADDR[31:0]=0x4 in order to be considered a hit.

If DISABLE_CHECK_SIZE is 1, then a read request will hit as long as the incoming ADDR[31:4]matches the expected_ADDR[31:4]. expected_ADDR[31:4] is always either "last ADDR[31:4]" or "last ADDR[31:4] + 1", where last ADDR is the prior read request actually issued by the AHB Master(as opposed to the last request to the CIF, which could have been a speculative read).

expected_ADDR[31:4] is always "last ADDR[31:4]" unless "SIZE" field in the last request uses the last byte in the 16-byte CIF word.

Offset: 0d8h | Read/Write: R/W | Reset: 0b0000

Bit	Reset	Description
3	0x0	DISABLE_CHECK_SIZE_MASTER4:
2	0x0	DISABLE_CHECK_SIZE_MASTER3:
1	0x0	DISABLE_CHECK_SIZE_MASTER2:
0	0x0	DISABLE_CHECK_SIZE_MASTER1:

10.4.2.2 AHB_ARBITRATION_XBAR_CTRL_0

XBAR Control Register

Offset: 0dch | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxx00

Bit	Reset	Description
16	0x0	MEM_INIT_DONE: SW 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 SW handles coherency then this can be turned off to improve performance.SW writes to modify 0 = ENABLE 1 = DISABLE
0	0x0	POST_DIS: SW writes to modify 0 = ENABLE 1 = DISABLE

10.4.2.3 AHB_AHB_MEM_PREFETCH_CFG3_0

See the description of the pre-fetcher above.

Offset: 0e0h | Read/Write: R/W | Reset: 0b00010100100xxxxx0000100000000000

Bit	R/W	Reset	Description
31	RW	0x0	ENABLE: 1=enable 0=disable
30:26	RW	0x5	AHB_MST_ID: AHBDMA master 0 = CPU 1 = COP 2 = VCP 3 = UNUSED_03 4 = IDE 5 = AHBDMA 6 = USB 7 = APBDMA 8 = XIO 9 = SDMMC1 10 = NAND_FLASH 11 = SNOR 12 = SDMMC4 13 = BSEV 14 = UNUSED_0E 15 = UNUSED_0F 12 = SDMMC4 16 = BSEA 17 = USB3 18 = USB2 19 = SDMMC2 20 = SDMMC3 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	RW	0x4	ADDR_BNDRY: 2^(n+4) byte boundary. any value >16 will use n=16
20:16	RO	X	SPEC_THROTTLE: not used for AP20 and beyond
15:0	RW	0x800	INACTIVITY_TIMEOUT: 2048 cycles

10.4.2.4 AHB_AHB_MEM_PREFETCH_CFG4_0

See the description of the pre-fetcher above.

Offset: 0e4h | Read/Write: R/W | Reset: 0b00010100100xxxxx0000100000000000

Bit	R/W	Reset	Description
31	RW	0x0	ENABLE: 1=enable 0=disable
30:26	RW	0x5	AHB_MST_ID: AHBDMA master 0 = CPU 1 = COP 2 = VCP 3 = UNUSED_03 4 = IDE 5 = AHBDMA 6 = USB 7 = APBDMA

Bit	R/W	Reset	Description
			8 = XIO 9 = SDMMC1 10 = NAND_FLASH 11 = SNOR 12 = SDMMC4 13 = BSEV 14 = UNUSED_0E 15 = UNUSED_0F 12 = SDMMC4 16 = BSEA 17 = USB3 18 = USB2 19 = SDMMC2 20 = SDMMC3 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	RW	0x4	ADDR_BNDRY: 2^(n+4) byte boundary. any value >16 will use n=16
20:16	RO	X	SPEC_THROTTLE: not used for AP20 and beyond
15:0	RW	0x800	INACTIVITY_TIMEOUT: 2048 cycles

10.4.2.5 AHB_AVP_PPCS_RD_COH_STATUS_0

Offset: 0e8h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxx

Bit	Reset	Description
16	X	RDS_OUTSTANDING
0	X	WRS_OUTSTANDING

10.4.2.6 AHB_AHB_MEM_PREFETCH_CFG1_0

See the description of the pre-fetcher above.

Offset: 0ech | Read/Write: R/W | Reset: 0b00010100100xxxxx0000100000000000

Bit	**R/W**	**Reset**	**Description**
31	RW	0x0	ENABLE: 1=enable 0=disable

Bit	R/W	Reset	Description
30:26	RW	0x5	AHB_MST_ID: AHBDMA master 0 = CPU 1 = COP 2 = VCP 3 = UNUSED_03 4 = IDE 5 = AHBDMA 6 = USB 7 = APBDMA 8 = XIO 9 = SDMMC1 10 = NAND_FLASH 11 = SNOR 12 = SDMMC4 13 = BSEV 14 = UNUSED_0E 15 = UNUSED_0F 12 = SDMMC4 16 = BSEA 17 = USB3 18 = USB2 19 = SDMMC2 20 = SDMMC3 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	RW	0x4	ADDR_BNDRY: 2^(n+4) byte boundary. any value >16 will use n=16
20:16	RO	X	SPEC_THROTTLE: not used for Tegra 200 Series and beyond
15:0	RW	0x800	INACTIVITY_TIMEOUT: 2048 cycles

10.4.2.7 AHB_AHB_MEM_PREFETCH_CFG2_0

See the description of the pre-fetcher above.

Offset: 0f0h | Read/Write: R/W | Reset: 0b00011000100xxxxx0000100000000000

Bit	R/W	Reset	Description
31	RW	0x0	ENABLE: 1=enable 0=disable

Bit	R/W	Reset	Description
30:26	RW	0x6	AHB_MST_ID: AHBDMA master 0 = CPU 1 = COP 2 = VCP 3 = UNUSED_03 4 = IDE 5 = AHBDMA 6 = USB 7 = APBDMA 8 = XIO 9 = SDMMC1 10 = NAND_FLASH 11 = SNOR 12 = SDMMC4 13 = BSEV 14 = UNUSED_0E 15 = UNUSED_0F 12 = SDMMC4 16 = BSEA 17 = USB3 18 = USB2 19 = SDMMC2 20 = SDMMC3 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	RW	0x4	ADDR_BNDRY: 2^(n+4) byte boundary. any value >16 will use n=16
20:16	RO	0x3	SPEC_THROTTLE: not used for Tegra 200 Series and beyond
15:0	RW	0x800	INACTIVITY_TIMEOUT:

10.4.2.8 AHB_AHBSLVMEM_STATUS_0

0x6000_C0F8: ahbslv outstanding rd, rdque_empty status

Offset: 0f4h | Read/Write: RO | Reset: 0bxx

Bit	Reset	Description
1	X	PPCS_RDS_OUTSTANDING
0	X	GIZMO_IP_RDQUE_EMPTY

10.4.2.9 AHB_ARBITRATION_AHB_MEM_WRQUE_MST_ID_0

AHB Memory Write Queue AHB Master ID Register

Offset: 0f8h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
30:0	0x0	AHB_MASTER_ID: 0 = there is no write data in the write queue from that AHB master.

10.4.2.10 AHB_ARBITRATION_CPU_ABORT_INFO_0

CPU Abort Info Register

Offset: 0fch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15	X	IRAMA: Abort occurred due to an iRAMa protection violation 0 = ABT_DIS 1 = ABT_EN
14	X	IRAMB: Abort occurred due to an iRAMb protection violation 0 = ABT_DIS 1 = ABT_EN
13	X	IRAMC: Abort occurred due to an iRAMc protection violation 0 = ABT_DIS 1 = ABT_EN
12	X	IRAMD: Abort occurred due to an iRAMd protection violation 0 = ABT_DIS 1 = ABT_EN
11	X	INV_IRAM: Abort occurred due to an access to invalid iRAM address space 0 = ABT_DIS 1 = ABT_EN
10	X	PPSB: Abort occurred due to a PPSB protection violation 0 = ABT_DIS 1 = ABT_EN
9	X	APB: Abort occurred due to an APB protection violation 0 = ABT_DIS 1 = ABT_EN
8	X	AHB: Abort occurred due to an AHB protection violation 0 = ABT_DIS 1 = ABT_EN
7	X	CACHE: Abort occurred due to a Cache protection violation 0 = ABT_DIS 1 = ABT_EN
6	X	PROTECTION: TRUE for any protection violation 0 = ABT_DIS 1 = ABT_EN

Bit	Reset	Description
5	X	ALIGN: TRUE for abort caused by Misalignment (i.e. word access at odd byte address) 0 = ABT_DIS 1 = ABT_EN
4	X	BADSIZE: TRUE for abort caused by Bad Size (i.e. word access at odd byte address) 0 = ABT_DIS 1 = ABT_EN
3	X	WRITE: Aborted transaction was a Write 0 = ABT_DIS 1 = ABT_EN
2	X	DATA: Aborted transaction was a Data access 0 = ABT_DIS 1 = ABT_EN
1:0	X	SIZE: Aborted transaction Request Size 00=byte, 01=hword, 10=word 0 = BYTE_ABT 1 = HWORD_ABT 2 = WORD_ABT

10.4.2.11 AHB_ARBITRATION_CPU_ABORT_ADDR_0

CPU Abort Address Register

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

Bit	Reset	Description
31:0	X	ADDR: Instruction Address which caused the abort

10.4.2.12 AHB_ARBITRATION_COP_ABORT_INFO_0

COP Abort Info Register

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

Bit	Reset	Description
15	X	IRAMA: Abort occurred due to an iRAMa protection violation 0 = ABT_DIS 1 = ABT_EN
14	X	IRAMB: Abort occurred due to an iRAMb protection violation 0 = ABT_DIS 1 = ABT_EN
13	X	IRAMC: Abort occurred due to an iRAMc protection violation 0 = ABT_DIS 1 = ABT_EN
10	X	PPSB: Abort occurred due to a PPSB protection violation 0 = ABT_DIS 1 = ABT_EN

Bit	Reset	Description
9	X	APB: Abort occurred due to an APB protection violation 0 = ABT_DIS 1 = ABT_EN
8	X	AHB: Abort occurred due to an AHB protection violation 0 = ABT_DIS 1 = ABT_EN
7	X	CACHE: Abort occurred due to a Cache protection violation 0 = ABT_DIS 1 = ABT_EN
6	X	PROTECTION: TRUE for any protection violation 0 = ABT_DIS 1 = ABT_EN
5	X	ALIGN: TRUE for abort caused by Misalignment (i.e. word access at odd byte address) 0 = ABT_DIS 1 = ABT_EN
4	X	BADSIZE: TRUE for abort caused by Bad Size (i.e. word access at odd byte address) 0 = ABT_DIS 1 = ABT_EN
3	X	WRITE: Aborted transaction was a Write 0 = ABT_DIS 1 = ABT_EN
2	X	DATA: Aborted transaction was a Data access 0 = ABT_DIS 1 = ABT_EN
1:0	X	SIZE: Aborted transaction Request Size 00=byte, 01=hword, 10=word 0 = BYTE_ABT 1 = HWORD_ABT 2 = WORD_ABT

10.4.2.13 AHB_ARBITRATION_COP_ABORT_ADDR_0

COP Abort Address Register

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

Bit	Reset	Description
31:0	X	ADDR: Instruction Address which caused the abort

10.4.2.14 AHB_AVPC_MCCIF_FIFOCTRL_0

The registers below allow optimizing the synchronization timing in the memory client asynchronous FIFOs. When they can be used depends on the client and memory controller clock ratio.

Additionally, the RDMC_RDFAST/RDCL_RDFAST fields can increase power consumption if the asynchronous FIFO is implemented as a real RAM. There is no power impact on latch-based FIFOs. Flipflop-based FIFOs do not use these fields.

Recommended Settings

Client writing to FIFO, memory controller reading from FIFO

- mcclk_freq <= clientclk_freq

You can enable both RDMC_RDFAST and WRCL_CLLE2X. If one of the FIFOs is a real RAM and power is a concern, avoid enabling RDMC_RDFAST.

- clientclk_freq < mcclk_freq <= 2 * clientclk_freq

You can enable RDMC_RDFAST or WRCL_MCLE2X, but because the client clock is slower, enable only WRCL_MCLE2X.

- 2 * clientclk_freq < mcclk_freq

You can only enable RDMC_RDFAST. If one of the FIFOs is a real RAM and power is a concern, avoid enabling RDMC_RDFAST.

Memory controller writing to FIFO, client reading from FIFO

- clientclk_freq <= mcclk_freq

You can enable both RDCL_RDFAST and WRMC_CLLE2X. If one of the FIFOs is a real RAM and power is a concern, avoid enabling RDCL_RDFAST.

- mcclk_freq < clientclk_freq <= 2 * mcclk_freq

You can enable RDCL_RDFAST or WRMC_CLLE2X, but because the memory controller clock is slower, enable only WRMC_CLLE2X.

- 2 * mcclk_freq < clientclk_freq

You can only enable RDCL_RDFAST. If one of the FIFOs is a real RAM and power is a concern, avoid enabling RDCL_RDFAST.

Memory Client Interface FIFO Control Register

Offset: 23ch | Read/Write: R/W | Reset: 0b0000

Bit	Reset	Description
3	DISABLE	AVPC_MCCIF_RDCL_RDFAST: 0 = DISABLE 1 = ENABLE
2	DISABLE	AVPC_MCCIF_WRMC_CLLE2X: 0 = DISABLE 1 = ENABLE
1	DISABLE	AVPC_MCCIF_RDMC_RDFAST: 0 = DISABLE 1 = ENABLE
0	DISABLE	AVPC_MCCIF_WRCL_MCLE2X: 0 = DISABLE 1 = ENABLE

10.4.2.15 AHB_TIMEOUT_WCOAL_AVPC_0

This register exists only for write clients. Reset value defaults to 50 for most clients, but may be different for certain clients.

Write coalescing happens inside the memory client. Coalescing means two (NV_MC_MW/2)-bit requests are grouped together in one NV_MC_MW-bit request. The register value indicates how many cycles a first write request is going to wait for a subsequent one for possible coalescing. The coalescing can only happen if the request addresses are compatible. A value of zero means that coalescing is off and requests are sent right away to the memory controller.

Write coalescing can have a very significant impact performance when accessing the internal memory, because its memory word is NV_MC_WM-bit wide. Grouping two half-word accesses is much more efficient, because the two accesses would actually have taken three cycles, due to a stall when accessing the same memory bank. It also reduces the number of accessing (one instead of two), freeing up internal memory bandwidth for other accesses.

The impact on external memory accesses is not as significant as the burst access is for NV_MC_MW/2 bits. But a coalesced write guarantees two consecutive same page accesses which is good for external memory bandwidth utilization.

The write coalescing time-out should be programmed depending on the client behavior. The first write is obviously delayed by an amount of client cycles equal to the time-out value.

Note:	Writes tagged by the client (i.e. the client expects a write response, usually for coherency), and the last write of a block transfer are not delayed. They only have a one-cycle opportunity to get coalesced.

Write Coalescing Time-Out Register

Offset: 240h | Read/Write: R/W | Reset: 0b00110010

Bit	Reset	Description
7:0	0x32	AVPCARM7W_WCOAL_TMVAL

10.5 AHB DMA Controller

The AHB DMA is a master placed between the AHB Bus and the Memory Controller (MC). It is used to accomplish block data transfers between external memory (DRAM) and internal memory (IRAM). It can also be used to move data from one section of external memory to another. It has four channels which can transfer data concurrently.

AHB DMA is used to transfer data from a source location to a destination location. Transfers can be from DRAM to DRAM, or between DRAM and IRAM in either direction. DMA transfers are done without any processor intervention other than the register writes to program the parameters for a particular transfer and handling interrupts.

Features

- DMA master for transfers between external memory and AHB
- Data can be transferred from DRAM to DRAM, DRAM to IRAM or IRAM to DRAM
- Two modes of data transfer: single transfer (once) or continuous
- Programmable burst sizes of 1, 4 or 8 words
- Max transfer size of 64 KB per channel
- Minimum transfer size is 4B per channel
- Separate source and destination address pointers
- Per channel trigger and flow control mechanism
- Channel to channel trigger support i.e. ability to start transfer from one channel upon completion of transfer of another channel
- Interrupts per channel can be routed to CPU or AVP
- Ability to hold processor until transfer is done
- Address strides and wrap modes supported in once mode
- Double-buffering mode which allows transfer of data to two sequential destination addresses
- Round robin arbitration among channels at burst granularity
- Runs on system clock

10.5.1 Functionality

There are 4 channels in AHB DMA. An AHB DMA channel can transfer specified portions of data from an AHB address space (basically used for IRAM) to an MC address space, or transfer data between two different locations of MC address space.MC and XMB are used interchangeably. AHB DMA follows a simple round robin arbitration scheme.

Each channel can have independent burst transfer sizes programmed to one word, four words, or eight words.

Each DMA channel supports:

- Enable bit (note that there is another enable bit known as global enable bit).
- Direction bit to determine the direction of transfer: AHB to MC or MC to AHB.
- Interrupt Enable at the end of transfer with ability to mask or route to desired processor.
- Ability to hold off a Processor. Processor which writes into this bit will be held until transfer is completed.
- Trigger and Flow selects. These are additional control bits on which the transfer depends; Trigger bit is used to start a channel on some event to start the transfer and Flow bit is used to proceed with the transfer on every burst. This again depends upon events from the system or AHB DMA itself. i.e., transfers can be started automatically, or under software control or under hardware control.
- Stride feature allows transfer of blocks of data where the transfer size and skip size are programmed.

- Wrap feature wraps the address on every burst transfer.
- Double Buffering Mode will make the AHB DMA Burst Address to reset to AHB Base Address after every even (2nd, 4th, 6th, etc) numbered transfer. Double buffering mode can be enabled with continuous transfer mode only.
- Separate registers for specifying AHB start address and MC start address.
- Ability to reduce the burst rate by delaying each burst by a programmable number of clock cycles.

10.5.2 Programming Guidelines

All the registers of a channel should be programmed before the channel enable bit is set.

Clearing the Global Enable bit causes the transfers that are in progress to be paused and setting the Global Enable resumes the transfers. If channel is disabled while transfer is in progress, transfer ends after completing any burst sequence that is in progress.

For best performance, MC source and destination addresses should be aligned to the burst size. For example, if Burst size is programmed to 4, the address needs to be aligned to 4-word boundary to achieve best throughput. If the MC address is not aligned to a 4-word boundary, 1-word transfers are initiated at the start and end of transfer as needed. Busy bit gets set as soon as a channel is enabled and gets cleared after transfer completes.

Interrupts are write-1-to-clear i.e., interrupt bit is cleared when the value of write data corresponding the bit position of the interrupt bit is 1.

10.5.3 AHB DMA Registers

10.5.3.1 AHBDMA_CMD_0

AHB-DMA Command Register

The Global Enable bit in the command register enables the AHB DMA. Clearing this bit causes active DMA transfers to be paused. Pending bus transactions (ongoing burst) are completed and no new transactions are generated. Set this bit again to resume transfers.

Offset: 000h | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
31	0x0	GEN: 0 = Disable AHB-DMA. 0 = DISABLE 1 = ENABLE

10.5.3.2 AHBDMA_STA_0

AHB-DMA Status Register

Busy bits in the status register indicate which (if any) of the AHB DMA channels have active pending AHB DMA transfers.

AHB DMA transfers that are started in continuous (repetitive) mode have their busy bits active until the enable bit for the AHB DMA channel is cleared to 0.

Offset: 004h | Read/Write: RO | Reset: 0bxxxx

Bit	Reset	Description
27	X	CH3: AHB DMA channel busy status flags set/cleared by HW 0 = NOT_BUSY 1 = BUSY

Bit	Reset	Description
26	X	CH2: AHB DMA channel busy status flags set/cleared by HW 0 = NOT_BUSY 1 = BUSY
25	X	CH1: AHB DMA channel busy status flags set/cleared by HW 0 = NOT_BUSY 1 = BUSY
24	X	CH0: AHB DMA channel busy status flags set/cleared by HW 0 = NOT_BUSY 1 = BUSY

10.5.3.3 AHBDMA_TX_REQ_0

AHB-DMA Requestor Assignments

The Tx Requestors are values used as triggers or flow controls for AHB DMA transfers.

Offset: 008h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	SMP_31: Software requestor SMP31 from the SHRD_SMP.STA register. 0 = DISABLE 1 = ENABLE
30	X	SMP_30: Software requestor SMP30 from the SHRD_SMP.STA register. 0 = DISABLE 1 = ENABLE
29	X	SMP_29: Software requestor SMP29 from the SHRD_SMP.STA register. 0 = DISABLE 1 = ENABLE
28	X	SMP_28: Software requestor SMP28 from the SHRD_SMP.STA register. 0 = DISABLE 1 = ENABLE
27	X	AHB_3: End of AHB-DMA Transfer on Channel 3 0 = DISABLE 1 = ENABLE
26	X	AHB_2: End of AHB-DMA Transfer on Channel 2 0 = DISABLE 1 = ENABLE
25	X	AHB_1: End of AHB-DMA Transfer on Channel 1 0 = DISABLE 1 = ENABLE
24	X	AHB_0: End of AHB-DMA Transfer on Channel 0 0 = DISABLE 1 = ENABLE
23	X	HRQ7_TMR2: Timer 2 Interrupt 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
22	X	HRQ6_TMR1: Timer 1 Interrupt 0 = DISABLE 1 = ENABLE
19	X	HRQ3_XRQ_D: 0 = NOP 0 = DISABLE 1 = ENABLE
18	X	HRQ2_XRQ_C: 0 = NOP 0 = DISABLE 1 = ENABLE
17	X	SMP_31_same_as_31_bit: Software requestor SMP31 from the SHRD_SMP.STA register (same as bit[31]) 0 = DISABLE 1 = ENABLE
16	X	SMP_30_same_as_30_bit: Software requestor SMP30 from the SHRD_SMP.STA register.(same as bit[30]) 0 = DISABLE 1 = ENABLE
15	X	SRQ1_XRQ_B: 0 = NOP 0 = DISABLE 1 = ENABLE
14	X	SRQ0_XRQ_A: 0 = NOP 0 = DISABLE 1 = ENABLE
11	X	Host1x: 0 = NOP 0 = DISABLE 1 = ENABLE
10	X	SMP_26: Enable software requestor SMP26 from the SHRD_SMP.STA register. 0 = DISABLE 1 = ENABLE
9	X	SMP_25: Enable software requestor SMP25 from the SHRD_SMP.STA register. 0 = DISABLE 1 = ENABLE
8	X	SMP_24: Enable software requestor SMP24 from the SHRD_SMP.STA register. 0 = DISABLE 1 = ENABLE
7	X	SMP_23: Enable software requestor SMP23 from the SHRD_SMP.STA register. 0 = DISABLE 1 = ENABLE
6	X	SMP_22: Enable software requestor SMP22 from the SHRD_SMP.STA register. 0 = DISABLE 1 = ENABLE
5	X	SMP_21: Enable software requestor SMP21 from the SHRD_SMP.STA register. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	X	SMP_20: Enable software requestor SMP20 from the SHRD_SMP.STA register. 0 = DISABLE 1 = ENABLE
3	X	SMP_19: Enable software requestor SMP19 from the SHRD_SMP.STA register. 0 = DISABLE 1 = ENABLE
2	X	SMP_18: Enable software requestor SMP18 from the SHRD_SMP.STA register. 0 = DISABLE 1 = ENABLE
1	X	SMP_17: Enable software requestor SMP17 from the SHRD_SMP.STA register. 0 = DISABLE 1 = ENABLE
0	X	CNTR_REQ: Enable Counter requestor. 0 = DISABLE 1 = ENABLE

10.5.3.4 AHBDMA_COUNTER_0

Controls which (if any) AHB DMA channel(s) will self-throttle based on the AHB DMA Counter Value. If any bit in [21:18] 4-bit field is active, then AHB DMA counter value will be enabled for decrement and reload (just as if CNTR.EN is set).

AHB-DMA Counter Usage

The AHB DMA Counter is used to slow down the request rates on some AHB DMA channels. It stores bits that configure which (if any) channel(s) should be throttled.

Offset: 010h | Read/Write: R/W | Reset: 0b00000000000000000000

Bit	Reset	Description
21	0x0	CH3_FL_CNT: Controls AHB DMA channel3 self-throttle based on the AHB DMA Counter Value. If this bit field is active, then AHB DMA counter value will be enabled for decrement and reload (just as if CNTR.EN is set). 0 = DISABLE 1 = ENABLE
20	0x0	CH2_FL_CNT: Controls AHB DMA channel2 self-throttle based on the AHB DMA Counter Value. If this bit field is active, then AHB DMA counter value will be enabled for decrement and reload (just as if CNTR.EN is set). 0 = DISABLE 1 = ENABLE
19	0x0	CH1_FL_CNT: Controls AHB DMA channel1 self-throttle based on the AHB DMA Counter Value. If this bit field is active, then AHB DMA counter value will be enabled for decrement and reload (just as if CNTR.EN is set). 0 = DISABLE 1 = ENABLE
18	0x0	CH0_FL_CNT: Controls AHB DMA channel0 self-throttle based on the AHB DMA Counter Value. If this bit field is active, then AHB DMA counter value will be enabled for decrement and reload (just as if CNTR.EN is set). 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
17	0x0	CNTR_EN: When this bit is set Counter is enabled. 0 = DISABLE 1 = ENABLE
16	0x0	PRD_EN: Normally, AHB DMA current count value is reloaded to the programmed init/reload value whenever the current count reaches 0. If this bit is set, the reload is additionally qualified with the condition that the last word of the burst is being sent. This will delay the reload of the counter by only a few cycles from when the counter value reaches 0. 0 = DISABLE 1 = ENABLE
15:0	0x0	COUNT_VALUE: DMA COUNT Init/Reload Value.

10.5.3.5 AHBDMA_IRQ_STA_CPU_0

AHB-DMA MASK_CPU Register Usage

Gathers all the after-masking CPU directed IRQ status bits.

Offset: 014h | Read/Write: RO | Reset: 0bxxxx

Bit	Reset	Description
3	X	CH3: Gathers all the after-masking CPU directed IRQ status bits from channel3 0 = DISABLE 1 = ENABLE
2	X	CH2: Gathers all the after-masking CPU directed IRQ status bits from channel2 0 = DISABLE 1 = ENABLE
1	X	CH1: Gathers all the after-masking CPU directed IRQ status bits from channel1 0 = DISABLE 1 = ENABLE
0	X	CH0: Gathers all the after-masking CPU directed IRQ status bits from channel0 0 = DISABLE 1 = ENABLE

10.5.3.6 AHBDMA_IRQ_STA_COP_0

AHB-DMA MASK_COP Register Usage

Gathers all the after-masking COP directed IRQ status bits.

Offset: 018h | Read/Write: RO | Reset: 0bxxxx

Bit	Reset	Description
3	X	CH3: Gathers all the after-masking COP directed IRQ status bits from channel3 0 = DISABLE 1 = ENABLE
2	X	CH2: Gathers all the after-masking COP directed IRQ status bits from channel2 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1	X	CH1: Gathers all the after-masking COP directed IRQ status bits from channel1 0 = DISABLE 1 = ENABLE
0	X	CH0: Gathers all the after-masking COP directed IRQ status bits from channel0 0 = DISABLE 1 = ENABLE

10.5.3.7 AHBDMA_IRQ_MASK_0

AHB-DMA SET_MASK Register Usage

Allows the IRQ to propagate when enabled, this is the ANDed result of set and clear mask bits.

Offset: 01ch | Read/Write: RO | Reset: 0bxxxx

Bit	Reset	Description
3	X	CH3: Each bit allows the associated channel3 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
2	X	CH2: Each bit allows the associated channel2 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
1	X	CH1: Each bit allows the associated channel1 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
0	X	CH0: Each bit allows the associated channel0 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE

10.5.3.8 AHBDMA_IRQ_MASK_SET_0

AHB-DMA CLR_MASK Register Usage

This register sets the Mask for the IRQs.

Offset: 020h | Read/Write: WO | Reset: 0b0000

Bit	Reset	Description
3	0x0	CH3: Writing 1 Sets the Mask Register for CH3 0 = DISABLE 1 = ENABLE
2	0x0	CH2: Writing 1 Sets the Mask Register for CH2 0 = DISABLE 1 = ENABLE
1	0x0	CH1: Writing 1 Sets the Mask Register for CH1 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
0	0x0	CH0: Writing 1 Sets the Mask Register for CH0 0 = DISABLE 1 = ENABLE

10.5.3.9 AHBDMA_IRQ_MASK_CLR_0

AHB-DMA Requestor Usage

This register clears the IRQs.

Offset: 024h | Read/Write: WO | Reset: 0b0000

Bit	Reset	Description
3	0x0	CH3: Writing 1 Clears the Mask Register for CH3 0 = DISABLE 1 = ENABLE
2	0x0	CH2: Writing 1 Clears the Mask Register for CH2 0 = DISABLE 1 = ENABLE
1	0x0	CH1: Writing 1 Clears the Mask Register for CH1 0 = DISABLE 1 = ENABLE
0	0x0	CH0: Writing 1 Clears the Mask Register for CH0 0 = DISABLE 1 = ENABLE

10.5.3.10 AHBDMA_RDWR_COHERENCY_0

0x6000_8028: AHBDMA outstanding R/W per channel

bit[31] is the coherency status on the AHB-side of the AHBDMA.

When MST_GIZMO_WRQUE_EMPTY is active, there are no writeAHB transactions from any AHBDMA channel on their way out.

bits[19:16] and [3:0] check the status of WriteXMB and ReadXMB transactions on the direct-to-memory CIF side of the AHBDMA.

Offset: 028h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	MST_GIZMO_WRQUE_EMPTY : When active, there are no writeAHB transactions
19	X	CH3_RDS_ACTIVE: 1=ahbdma channel 3 has read transaction(s) outstanding
18	X	CH2_RDS_ACTIVE
17	X	CH1_RDS_ACTIVE
16	X	CH0_RDS_ACTIVE
3	X	CH3_WRS_ACTIVE: 1=ahbdma channel 3 has write transaction(s) outstanding

Bit	Reset	Description
2	X	CH2_WRS_ACTIVE
1	X	CH1_WRS_ACTIVE
0	X	CH0_WRS_ACTIVE

10.5.3.11 AHBDMA_TEST_BUS_0

0x6000_802C: AHBDMA_TEST_BUS. This is a legacy observability bus.

Offset: 02ch | Read/Write: RO | Reset: 0bxxxxxxxx

Bit	Reset	Description
7	X	MSTGIZMO_WRQUE_CH3_PONG_STATUS
6	X	MSTGIZMO_WRQUE_CH3_PING_STATUS
5	X	MSTGIZMO_WRQUE_CH2_PONG_STATUS
4	X	MSTGIZMO_WRQUE_CH2_PING_STATUS
3	X	MSTGIZMO_WRQUE_CH1_PONG_STATUS
2	X	MSTGIZMO_WRQUE_CH1_PING_STATUS
1	X	MSTGIZMO_WRQUE_CH0_PONG_STATUS
0	X	MSTGIZMO_WRQUE_CH0_PING_STATUS

10.5.3.12 AHBDMA_PPCS_MCCIF_FIFOCTRL_0

Memory Client Interface FIFO Control Register

The registers below allow optimizing the synchronization timing in the memory client asynchronous FIFOs. When they can be used depend on the client and memory controller clock ratio.

Additionally, the RDMC_RDFAST/RDCL_RDFAST fields can increase power consumption if the asynchronous FIFO is implemented as a real ram.

There is no power impact on latch-based FIFOs. Flip-flop-based FIFOs do not use these fields.

Recommended Settings

Client writing to FIFO, memory controller reading from FIFO

mcclk_freq <= clientclk_freq

You can enable both RDMC_RDFAST and WRCL_CLLE2X. If one of the FIFOs is a real ram and power is a concern, you should avoid enabling RDMC_RDFAST.

clientclk_freq < mcclk_freq <= 2 * clientclk_freq

You can enable RDMC_RDFAST or WRCL_MCLE2X, but because the client clock is slower, you should enable only WRCL_MCLE2X.

2 * clientclk_freq < mcclk_freq

You can only enable RDMC_RDFAST. If one of the FIFOs is a real ram and power is a concern, you should avoid enabling RDMC_RDFAST.

Memory controller writing to fifo, client reading from fifo

clientclk_freq <= mcclk_freq

You can enable both RDCL_RDFAST and WRMC_CLLE2X. If one of the FIFOs is a real ram and power is a concern, you should avoid enabling RDCL_RDFAST.

mcclk_freq < clientclk_freq <= 2 * mcclk_freq

You can enable RDCL_RDFAST or WRMC_CLLE2X, but because the memory controller clock is slower, you should enable only WRMC_CLLE2X.

2 * mcclk_freq < clientclk_freq

You can only enable RDCL_RDFAST. If one of the FIFOs is a real ram and power is a concern, you should avoid enabling RDCL_RDFAST.

Offset: 040h | Read/Write: R/W | Reset: 0b0000

Bit	Reset	Description
3	0x0	PPCS_MCCIF_RDCL_RDFAST: 0 = DISABLE 1 = ENABLE
2	0x0	PPCS_MCCIF_WRMC_CLLE2X: 0 = DISABLE 1 = ENABLE
1	0x0	PPCS_MCCIF_RDMC_RDFAST: 0 = DISABLE 1 = ENABLE
0	0x0	PPCS_MCCIF_WRCL_MCLE2X: 0 = DISABLE 1 = ENABLE

10.5.3.13 AHBDMA_TIMEOUT_WCOAL_PPCS_0

Write Coalescing Time-Out Register

This register exists only for write clients. Reset value defaults to 50 for most clients, but may be different for certain clients.

Write coalescing happens inside the memory client.

Coalescing means two (NV_MC_MW/2)-bit requests are grouped together in one NV_MC_MW-bit request.

The register value indicates how many cycles a first write request is going to wait for a subsequent one for possible coalescing. The coalescing can only happen if the request addresses are compatible. A value of zero means that coalescing is off and requests are sent right away to the memory controller.

Write coalescing can have a very significant impact performance when accessing the internal memory, because its memory word is NV_MC_WM-bit wide. Grouping two half-word accesses is much more efficient, because the two accesses would actually have taken three cycles, due to a stall when accessing the same memory bank. It also reduces the number of accessing (one instead of two), freeing up internal memory bandwidth for other accesses.

The impact on external memory accesses is not as significant as the burst access is for NV_MC_MW/2 bits. But a coalesced write guarantees two consecutive same page accesses which is good for external memory bandwidth utilization.

The write coalescing time-out should be programmed depending on the client behavior. The first write is obviously delayed by an amount of client cycles equal to the time-out value.

Note that writes tagged by the client (i.e. the client expects a write response, usually for coherency), and the last write of a block transfer are not delayed. They only have a one-cycle opportunity to get coalesced.

Offset: 044h | Read/Write: R/W | Reset: 0b0011001000110010

Bit	Reset	Description
15:8	0x32	PPCSAHBDMAW_WCOAL_TMVAL
7:0	0x32	PPCSAHBSLVW_WCOAL_TMVAL

10.5.4 AHBDMACHANNEL_REGISTERS

The description given for AHB-DMA-CHANNEL-0 applies to all the corresponding registers for the remaining channels AHB-DMA-CHANNEL-1 to AHB-DMA-CHANNEL-3

10.5.4.1 AHBDMACHAN_CHANNEL_0_CSR_0

AHB-DMA-CHANNEL-0 Control Register

Writing a 1 to bit [31] of an AHB DMA Channel Control Register will initiate the AHB DMA Transfer. Because this action will depend on some values programmed in the other registers, it is recommended that the registers in the address space in the required AHB DMA channel be programmed before writing into control register.

In "Once" mode, the channel is disabled after the AHB DMA Transfer has completed. When the channel's Transfer completes, an interrupt will be sent if the IE.EOC bit is set

If the transfer requires a trigger or flow, then the corresponding trigger selected by the "TRIG_SEL" or "REQ_SEL" field must become active respectively before the channel can starts its transfer. If both Trigger and Flow bits are set, both conditions must be met before the channel starts each DMA Burst.

Offset: 000h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: 0 = Disable the DMA Channel 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: 0 = NOP or waiting 0 = DISABLE 1 = ENABLE
29	0x0	FL_BTWN: 0 = do not flush 0 = DISABLE 1 = ENABLE
28	0x0	HOLD: 0 = Disable 0 = DISABLE 1 = ENABLE
27	0x0	DIR: 0 = XMB read to AHB write 0 = DISABLE 1 = ENABLE
26	0x0	ONCE: 0 = Run for Multiple Block Transfer 0 = DISABLE 1 = ENABLE
25	0x0	TRIG: 0 = Independent of Trigger (no trigger set) 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
24	0x0	FLOW: 0 = Independent of DRQ request (no flow set) 0 = DISABLE 1 = ENABLE
23:20	0x0	TRIG_SEL: 0 = SMP_30_same_as_bit_14 1 = SMP_31_same_as_bit_15 2 = HRQ2_XRQ_C 3 = HRQ3_XRQ_D 4 = HRQ4_N_A 5 = HRQ5_N_A 6 = HRQ6_TMR1 7 = HRQ7_TMR2 8 = AHB_0 9 = AHB_1 10 = AHB_2 11 = AHB_3 12 = SMP_28 13 = SMP_29 14 = SMP_30 15 = SMP_31
19:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = SMP_17 2 = SMP_18 3 = SMP_19 4 = SMP_20 5 = SMP_21 6 = SMP_22 7 = SMP_23 8 = SMP_24 9 = SMP_25 10 = SMP_26 11 = Host1x 12 = SRQ0_N_A 13 = SRQ1_N_A 14 = SRQ0_XRQ_A 15 = SRQ1_XRQ_B
15:2	0x0	WCOUNT: Number of 32 bit word cycles. This is encoded as N+1, so that if a value of 0 is programmed here, 1 32-bit word will be transferred.

10.5.4.2 AHBDMACHAN_CHANNEL_0_STA_0

AHB-DMA-CHANNEL-0 Status Register

Offset: 004h | Read/Write: R/W | Reset: 0b00x0xxxxxxxxxxx00000000000000

Bit	Reset	Description
31	0x0	BSY: 0 = NOP Read-only 0 = DISABLE 1 = ENABLE
30	0x0	IS_EOC: Write '1' to clear the flag 0 = NO_INTR 1 = INTR

Bit	Reset	Description
28	0x0	HALT: 0 = NOP (holding status) Read-only 0 = DISABLE 1 = ENABLE
15:2	0x0	COUNT: Remaining transfer count in 32bit word cycles Flags set / cleared by HW.

10.5.4.3 AHBDMACHAN_CHANNEL_0_AHB_PTR_0

AHB-DMA-CHANNEL-0 AHB Starting Address Pointer Register

If stride is enabled, the AHB address should be aligned to burst size.

Offset: 010h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: AHB-DMA starting address for internal AHB Bus

10.5.4.4 AHBDMACHAN_CHANNEL_0_AHB_SEQ_0

AHB-DMA-CHANNEL-0 AHB Address Sequencer Register

An AHB DMA stride is a sub-section of an AHB DMA transfer, can be used in once mode, and is composed of one or more AHB DMA bursts.

When stride is enabled, wrap should not be enabled. The size of a stride is programmable, but is always an integer multiple of 4 words.

The size of the AHB DMA stride must always be an integer multiple of the AHB DMA burst size.

The size of the AHB DMA transfer count (total words in the transfer) must also be a multiple of both the burst size and the stride size.

If stride is enabled, the AHB_STRIDE should be aligned to burst size.

Offset: 014h | Read/Write: R/W | Reset: 0b0xxx0010000x000000000000000000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 1 = CPU 0 = COP
27	0x0	AHB_DATA_SWAP: 0 = no data conversion, when 1 and writeAhb: [31:0]->{[7:0],[15:8],[23:16],[31:24]} 0 = DISABLE 1 = ENABLE
26:24	0x2	AHB_BURST: 0,1,5-7 = rsvd 2 = WORD 3 = FOUR_WORD 4 = EIGHT_WORD
23:21	0x0	AHB_STRIDE_SIZE: 0 = disabled, otherwise stride is enabled 0 = DISABLE 1 = WORDS_4 2 = WORDS_8 3 = WORDS_16 4 = WORDS_32 5 = WORDS_64 6 = WORDS_128 7 = WORDS_256

Bit	Reset	Description
19	0x0	DBL_BUF: 0 = Reload Base Address for 1X blocks (default), reload each time 0 = DISABLE 1 = ENABLE
18	0x0	AHB_ADDR_WRAP: 0 = disable 0 = DISABLE 1 = ENABLE
17:2	0x0	AHB_STRIDE: This field tells us number of words that are supposed to be skipped on AHB side.

10.5.4.5 AHBDMACHAN_CHANNEL_0_XMB_PTR_0

AHB-DMA-CHANNEL-0 XMB Starting Address Pointer Register

The starting XMB address for the AHB DMA transfer must be aligned to an address boundary that is a multiple of the burst size.

Offset: 018h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	XMB_BASE: AHB-DMA Starting address for internal AHB Bus

10.5.4.6 AHBDMACHAN_CHANNEL_0_XMB_SEQ_0

AHB-DMA-CHANNEL-0 XMB Address Sequencer Register

If the stride is enabled, then the XMB_STRIDE must also be a multiple of the burst size.

Offset: 01ch | Read/Write: R/W | Reset: 0b0xxxxxxx000000000000000000

Bit	Reset	Description
27	0x0	XMB_DATA_SWAP: 0 = no data conversion, when 1 and writeXmb: [31:0]->{[7:0],[15:8],[23:16],[31:24]} 0 = DISABLE 1 = ENABLE
19	0x0	DBL_BUF: 0 = Reload Base Address for 1X blocks (default) (reload each time) 0 = DISABLE 1 = ENABLE
18	0x0	XMB_ADDR_WRAP: 0 disable 0 = DISABLE 1 = ENABLE
17:2	0x0	XMB_STRIDE: This field tells us number of words that are supposed to be skipped on XMB side.

10.5.4.7 AHBDMACHAN_CHANNEL_1_CSR_0

AHB-DMA-CHANNEL-1 Control Register

Offset: 020h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: 0 = Disable the DMA Channel 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
30	0x0	IE_EOC: 0 = NOP 0 = DISABLE 1 = ENABLE
29	0x0	FL_BTWN: 0 = do not flush 0 = DISABLE 1 = ENABLE
28	0x0	HOLD: 0 = Disable 0 = DISABLE 1 = ENABLE
27	0x0	DIR: 0 = XMB read to AHB write 0 = DISABLE 1 = ENABLE
26	0x0	ONCE: 0 = Run for Multiple Block Transfer 0 = DISABLE 1 = ENABLE
25	0x0	TRIG: 0 = Independent of Trigger 0 = DISABLE 1 = ENABLE
24	0x0	FLOW: 0 = Independent of DRQ request 0 = DISABLE 1 = ENABLE
23:20	0x0	TRIG_SEL: 0 = SMP_30_same_as_bit_14 1 = SMP_31_same_as_bit_15 2 = HRQ2_XRQ_C 3 = HRQ3_XRQ_D 4 = HRQ4_N_A 5 = HRQ5_N_A 6 = HRQ6_TMR1 7 = HRQ7_TMR2 8 = AHB_0 9 = AHB_1 10 = AHB_2 11 = AHB_3 12 = SMP_28 13 = SMP_29 14 = SMP_30 15 = SMP_31
19:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = SMP_17 2 = SMP_18 3 = SMP_19 4 = SMP_20 5 = SMP_21 6 = SMP_22 7 = SMP_23 8 = SMP_24 9 = SMP_25 10 = SMP_26 11 = Host1x 12 = SRQ0_N_A 13 = SRQ1_N_A 14 = SRQ0_XRQ_A 15 = SRQ1_XRQ_B
15:2	0x0	WCOUNT: Number of 32 bit word cycles. This is encoded as N+1, so that if a value of 0 is programmed here, 1 32-bit word will be transferred.

10.5.4.8 AHBDMACHAN_CHANNEL_1_STA_0

AHB-DMA-CHANNEL-1 Status Register

Offset: 024h | Read/Write: R/W | Reset: 0b00x0xxxxxxxxxxx00000000000000

Bit	Reset	Description
31	0x0	BSY: 0 = NOP Read-only 0 = DISABLE 1 = ENABLE
30	0x0	IS_EOC: Write '1' to clear the flag 0 = NO_INTR 1 = INTR
28	0x0	HALT: 0 = NOP 0 = DISABLE 1 = ENABLE
15:2	0x0	COUNT: Remaining transfer count in 32bit word cycles Flags set / cleared by HW

10.5.4.9 AHBDMACHAN_CHANNEL_1_AHB_PTR_0

AHB-DMA-CHANNEL-1 AHB Starting Address Pointer Register

Offset: 030h | Read/Write: R/W |Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: AHB-DMA starting address pointer for internal AHB Bus

10.5.4.10 AHBDMACHAN_CHANNEL_1_AHB_SEQ_0

AHB-DMA-CHANNEL-1 AHB Address Sequencer Register

Offset: 034h | Read/Write: R/W | Reset: 0b0xxx0010000x000000000000000000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 1 = CPU 0 = COP
27	0x0	AHB_DATA_SWAP: 0 = no data conversion, when 1 and writeAhb: [31:0]->{[7:0],[15:8],[23:16],[31:24]} enum (DISABLE=0x0,ENABLE=0x1)
26:24	0x2	AHB_BURST: 0,1,5-7 = rsvd 2 = WORD 3 = FOUR_WORD 4 = EIGHT_WORD
23:21	0x0	AHB_STRIDE_SIZE: 0 = disabled, otherwise stride is enabled 0 = DISABLE 1 = WORDS_4 2 = WORDS_8 3 = WORDS_16 4 = WORDS_32 5 = WORDS_64 6 = WORDS_128 7 = WORDS_256
19	0x0	DBL_BUF: 0 = Reload Base Address for 1X blocks (default), reload each time 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
18	0x0	AHB_ADDR_WRAP: 0 = disable 0 = DISABLE 1 = ENABLE
17:2	0x0	AHB_STRIDE: This field tells us number of words that are supposed to be skipped on AHB side.

10.5.4.11 AHBDMACHAN_CHANNEL_1_XMB_PTR_0

AHB-DMA-CHANNEL-1 XMB Starting Address Pointer Register

Offset: 038h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	XMB_BASE: AHB-DMA Starting address pointer for internal AHB Bus

10.5.4.12 AHBDMACHAN_CHANNEL_1_XMB_SEQ_0

AHB-DMA-CHANNEL-1 XMB Address Sequencer Register

Offset: 03ch | Read/Write: R/W | Reset: 0b0xxxxxxx000000000000000000

Bit	Reset	Description
27	0x0	XMB_DATA_SWAP: 0 = no data conversion, when 1 and writeXmb: [31:0]->{[7:0],[15:8],[23:16],[31:24]} 0 = DISABLE 1 = ENABLE
19	0x0	DBL_BUF: 0 = Reload Base Address for 1X blocks (default), reload each time 0 = DISABLE 1 = ENABLE
18	0x0	XMB_ADDR_WRAP: 1 enable: 0 disable ; When enabled the Address on XMB gets wrapped to same address. 0 = DISABLE 1 = ENABLE
17:2	0x0	XMB_STRIDE: This field tells us number of words that are supposed to be skipped on XMB side.

10.5.4.13 AHBDMACHAN_CHANNEL_2_CSR_0

AHB-DMA-CHANNEL-2 Control Register

Offset: 040h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: 0 = Disable the DMA Channel 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: 0 = NOP or waiting 0 = DISABLE 1 = ENABLE
29	0x0	FL_BTWN: 0 = do not flush 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
28	0x0	HOLD: 0 = Disable 0 = DISABLE 1 = ENABLE
27	0x0	DIR: 0 = XMB read to AHB write 0 = DISABLE 1 = ENABLE
26	0x0	ONCE: 0 = Run for Multiple Block Transfer 0 = DISABLE 1 = ENABLE
25	0x0	TRIG: 0 = Independent of Trigger (no trigger set) 0 = DISABLE 1 = ENABLE
24	0x0	FLOW: 0 = Independent of DRQ request (no flow set) 0 = DISABLE 1 = ENABLE
23:20	0x0	TRIG_SEL: 0 = SMP_30_same_as_bit_14 1 = SMP_31_same_as_bit_15 2 = HRQ2_XRQ_C 3 = HRQ3_XRQ_D 4 = HRQ4_N_A 5 = HRQ5_N_A 6 = HRQ6_TMR1 7 = HRQ7_TMR2 8 = AHB_0 9 = AHB_1 10 = AHB_2 11 = AHB_3 12 = SMP_28 13 = SMP_29 14 = SMP_30 15 = SMP_31
19:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = SMP_17 2 = SMP_18 3 = SMP_19 4 = SMP_20 5 = SMP_21 6 = SMP_22 7 = SMP_23 8 = SMP_24 9 = SMP_25 10 = SMP_26 11 = Host1x 12 = SRQ0_N_A 13 = SRQ1_N_A 14 = SRQ0_XRQ_A 15 = SRQ1_XRQ_B
15:2	0x0	WCOUNT: Number of 32 bit word cycles. This is encoded as N+1, so that if a value of 0 is programmed here, 1 32-bit word will be transferred.

10.5.4.14 AHBDMACHAN_CHANNEL_2_STA_0

AHB-DMA-CHANNEL-2 Status Register

Offset: 044h | Read/Write: R/W | Reset: 0b00x0xxxxxxxxxxx00000000000000

Bit	Reset	Description
31	0x0	BSY: 0 = NOP Read-only 0 = DISABLE 1 = ENABLE
30	0x0	IS_EOC: Write '1' to clear the flag 0 = NO_INTR 1 = INTR
28	0x0	HALT: 0 = NOP (holding status) Read-only 0 = DISABLE 1 = ENABLE
15:2	0x0	COUNT: Remaining transfer count in 32bit word cycles Flags set / cleared by HW.

10.5.4.15 AHBDMACHAN_CHANNEL_2_AHB_PTR_0

AHB-DMA-CHANNEL-2 AHB Starting Address Pointer Register

Offset: 050h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: AHB-DMA starting address for internal AHB Bus

10.5.4.16 AHBDMACHAN_CHANNEL_2_AHB_SEQ_0

AHB-DMA-CHANNEL-2 AHB Address Sequencer Register

Offset: 054h | Read/Write: R/W | Reset: 0b0xxx0010000x000000000000000000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 1 = CPU 0 = COP
27	0x0	AHB_DATA_SWAP: 0 = no data conversion, when 1 and writeAhb: [31:0]->{[7:0],[15:8],[23:16],[31:24]} 0 = DISABLE 1 = ENABLE
26:24	0x2	AHB_BURST: 0,1,5-7 = rsvd 2 = WORD 3 = FOUR_WORD 4 = EIGHT_WORD
23:21	0x0	AHB_STRIDE_SIZE: 0 = disabled, otherwise stride is enabled 0 = DISABLE 1 = WORDS_4 2 = WORDS_8 3 = WORDS_16 4 = WORDS_32 5 = WORDS_64 6 = WORDS_128 7 = WORDS_256

Bit	Reset	Description
19	0x0	DBL_BUF: 0 = Reload Base Address for 1X blocks (default), reload each time 0 = DISABLE 1 = ENABLE
18	0x0	AHB_ADDR_WRAP: 0 = disable 0 = DISABLE 1 = ENABLE
17:2	0x0	AHB_STRIDE: This field tells us number of words that are supposed to be skipped on AHB side.

10.5.4.17 AHBDMACHAN_CHANNEL_2_XMB_PTR_0

AHB-DMA-CHANNEL-2 XMB Starting Address Pointer Register

Offset: 058h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	XMB_BASE: AHB-DMA Starting address for internal AHB Bus

10.5.4.18 AHBDMACHAN_CHANNEL_2_XMB_SEQ_0

AHB-DMA-CHANNEL-2 XMB Address Sequencer Register

Offset: 05ch | Read/Write: R/W | Secure: Unprotected | Reset: 0b0xxxxxxx000000000000000000 | Default: 0000.0000

Bit	Reset	Description
27	0x0	XMB_DATA_SWAP: 0 = no data conversion, when 1 and writeXmb: [31:0]->{[7:0],[15:8],[23:16],[31:24]} 0 = DISABLE 1 = ENABLE
19	0x0	DBL_BUF: 0 = Reload Base Address for 1X blocks (def) eload each time) 0 = DISABLE 1 = ENABLE
18	0x0	XMB_ADDR_WRAP: 1 enable: 0 disable ; When enabled the Address on XMB gets wrapped to same address. 0 = DISABLE 1 = ENABLE
17:2	0x0	XMB_STRIDE: This field tells us number of words that are supposed to be skipped on XMB side.

10.5.4.19 AHBDMACHAN_CHANNEL_3_CSR_0

AHB-DMA-CHANNEL-3 Control Register

Offset: 060h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: 0 = Disable the DMA Channel 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: 0 = NOP 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
29	0x0	FL_BTWN: 0 = do not flush 0 = DISABLE 1 = ENABLE
28	0x0	HOLD: 0 = Disable 0 = DISABLE 1 = ENABLE
27	0x0	DIR: 0 = XMB read to AHB write 0 = DISABLE 1 = ENABLE
26	0x0	ONCE: 0 = Run for Multiple Block Transfer 0 = DISABLE 1 = ENABLE
25	0x0	TRIG: 0 = Independent of Trigger 0 = DISABLE 1 = ENABLE
24	0x0	FLOW: 0 = Independent of DRQ request 0 = DISABLE 1 = ENABLE
23:20	0x0	TRIG_SEL: 0 = SMP_30_same_as_bit_14 1 = SMP_31_same_as_bit_15 2 = HRQ2_XRQ_C 3 = HRQ3_XRQ_D 4 = HRQ4_N_A 5 = HRQ5_N_A 6 = HRQ6_TMR1 7 = HRQ7_TMR2 8 = AHB_0 9 = AHB_1 10 = AHB_2 11 = AHB_3 12 = SMP_28 13 = SMP_29 14 = SMP_30 15 = SMP_31
19:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = SMP_17 2 = SMP_18 3 = SMP_19 4 = SMP_20 5 = SMP_21 6 = SMP_22 7 = SMP_23 8 = SMP_24 9 = SMP_25 10 = SMP_26 11 = Host1x 12 = SRQ0_N_A 13 = SRQ1_N_A 14 = SRQ0_XRQ_A 15 = SRQ1_XRQ_B
19:16	0x0	REQ_SEL: Lower 16 Requestors (15-0)
15:2	0x0	WCOUNT: Number of 32 bit word cycles. This is encoded as N+1, so that if a value of 0 is programmed here, 1 32-bit word will be transferred.

10.5.4.20 AHBDMACHAN_CHANNEL_3_STA_0

AHB-DMA-CHANNEL-3 Status Register

Offset: 064h | Read/Write: R/W | Reset: 0b00x0xxxxxxxxxxx00000000000000

Bit	Reset	Description
31	0x0	BSY: 0 = NOP Read-only 0 = DISABLE 1 = ENABLE
30	0x0	IS_EOC: Write '1' to clear the flag 0 = NO_INTR 1 = INTR
28	0x0	HALT: 0 = NOP 0 = DISABLE 1 = ENABLE
15:2	0x0	COUNT: Remaining transfer count in 32bit word cycles Flags set / cleared by HW

10.5.4.21 AHBDMACHAN_CHANNEL_3_AHB_PTR_0

AHB-DMA-CHANNEL-3 AHB Starting Address Pointer Register

Offset: 070h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: AHB-DMA starting address pointer for internal AHB Bus

10.5.4.22 AHBDMACHAN_CHANNEL_3_AHB_SEQ_0

AHB-DMA-CHANNEL-3 AHB Address Sequencer Register

Offset: 074h | Read/Write: R/W | Reset: 0b0xxx0010000x000000000000000000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 1 = CPU 0 = COP
27	0x0	AHB_DATA_SWAP: 0 = no data conversion, when 1 and writeAhb: [31:0]->{[7:0],[15:8],[23:16],[31:24]} 0 = DISABLE 1 = ENABLE
26:24	0x2	AHB_BURST: 0,1,5-7 = rsvd 2 = WORD 3 = FOUR_WORD 4 = EIGHT_WORD
23:21	0x0	AHB_STRIDE_SIZE: 0 = disabled, otherwise stride is enabled 0 = DISABLE 1 = WORDS_4 2 = WORDS_8 3 = WORDS_16 4 = WORDS_32 5 = WORDS_64 6 = WORDS_128 7 = WORDS_256
19	0x0	DBL_BUF: 0 = Reload Base Address for 1X blocks (default), reload each time 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
18	0x0	AHB_ADDR_WRAP: 0 = disable 0 = DISABLE 1 = ENABLE
17:2	0x0	AHB_STRIDE: This field tells us number of words that are supposed to be skipped on AHB side.

10.5.4.23 AHBDMACHAN_CHANNEL_3_XMB_PTR_0

AHB-DMA-CHANNEL-3 XMB Starting Address Pointer Register

Offset: 078h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	XMB_BASE: AHB-DMA Starting address pointer for internal AHB Bus

10.5.4.24 AHBDMACHAN_CHANNEL_3_XMB_SEQ_0

AHB-DMA-CHANNEL-3 XMB Address Sequencer Register

Offset: 07ch | Read/Write: R/W | Reset: 0b0xxxxxxx000000000000000000

Bit	Reset	Description
27	0x0	XMB_DATA_SWAP: 0 = no data conversion, when 1 and writeXmb: [31:0]->{[7:0],[15:8],[23:16],[31:24]} 0 = DISABLE 1 = ENABLE
19	0x0	DBL_BUF: 0 = Reload Base Address for 1X blocks (default), reload each time 0 = DISABLE 1 = ENABLE
18	0x0	XMB_ADDR_WRAP: 0 = disable 0 = DISABLE 1 = ENABLE
17:2	0x0	XMB_STRIDE: This field tells us number of words that are supposed to be skipped on XMB side.

11.0 APB

11.1 APB Miscellaneous Registers

11.1.1 Strap Registers

This register allows the state of the HW straps on the board to be read. These strap values are captured at reset.

11.1.1.1 APB_MISC_PP_STRAPPING_OPT_A_0

Strapping Options Register

Offset: 008h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx0xxxxxx0

Bit	Reset	Description
29:26	X	BOOT_SELECT: read at power-on reset time from GMI_AD[15:12] strap pads
25	X	BOOT_SRC_USB_RECOVERY_MODE: read at power-on reset time from GMI_HIOR strap pad 0 = DISABLED 1 = ENABLED
24	X	BOOT_SRC_NOR_BOOT: read at power-on reset time from GMI_HIOW strap pad 0 = IROM 1 = NOR
23:22	X	ARM_JTAG: read at power-on reset time from {GMI_CLK, GMI_ADV_N} strap pads 00=Serial_JTAG, 01=CPU_only, 10=COP_only, 11=Serial_JTAG (same as 00 case) 0 = SERIAL 1 = CPU 2 = COP 3 = SERIAL_ALT
8	RSVD1	MIO_WIDTH: 0 = RSVD1 1 = RSVD2
7:4	X	RAM_CODE: read at power-on reset time from GMI_AD[7:4] strap pads. In emulation (HIDREV_MAJORREV==0), this field indicates the RAM type connected. For QT (HIDREV_MINORREV==0): 0=SIM, 1=DDR, 2=DDR2, 3=LPDDR2 For FPGA (HIDREV_MINORREV==1): 0=SIM, 1=DDR, 2=DDR2, 3=LPDDR2
0	RSVD1	NOR_WIDTH: 0 = IS16BIT 1 = IS8BIT

11.1.2 Pad Tri-State Control Registers

11.1.2.1 APB_MISC_PP_TRISTATE_REG_A_0

These registers are used to control the PAD behavior. The TRISTATE registers should be programmed before tri-state pads which are not used for Power consumptions. Z_ indicates the enable for TRISTATING the pads, Enum TRISTATE indicates Outputs are Tristate, and NORMAL indicates they are Not Tristated.

TRI STATES THE Output Pads when the Tristate Bit is enabled. This is used by SW for Power Consumption.

0 = Normal; 1 = tristate Output.

TRI_STATE Configuration Register

Offset: 014h | Read/Write: R/W | Reset: 0b11000000000110111111111111110000

Bit	Reset	Description
31	TRISTATE	Z_OWC: 0 = NORMAL 1 = TRISTATE
30	TRISTATE	Z_SDIO1: 0 = NORMAL 1 = TRISTATE
29	NORMAL	Z_GMC: 0 = NORMAL 1 = TRISTATE
28	NORMAL	Z_GMA: 0 = NORMAL 1 = TRISTATE
27	NORMAL	Z_KBCF: 0 = NORMAL 1 = TRISTATE
26	NORMAL	Z_KBCE: 0 = NORMAL 1 = TRISTATE
25	NORMAL	Z_RM: 0 = NORMAL 1 = TRISTATE
24	TRISTATE	Z_PTA: 0 = NORMAL 1 = TRISTATE
23	NORMAL	Z_PMC: 0 = NORMAL 1 = TRISTATE
22	NORMAL	Z_KBCA: 0 = NORMAL 1 = TRISTATE
21	NORMAL	Z_KBCB: 0 = NORMAL 1 = TRISTATE
20	TRISTATE	Z_IRRX: 0 = NORMAL 1 = TRISTATE
19	TRISTATE	Z_IRTX: 0 = NORMAL 1 = TRISTATE
18	NORMAL	Z_I2CP: 0 = NORMAL 1 = TRISTATE

Bit	Reset	Description
17	TRISTATE	Z_GPV: 0 = NORMAL 1 = TRISTATE
16	TRISTATE	Z_GPU: 0 = NORMAL 1 = TRISTATE
15	TRISTATE	Z_DTE: 0 = NORMAL 1 = TRISTATE
14	TRISTATE	Z_DTD: 0 = NORMAL 1 = TRISTATE
13	TRISTATE	Z_DTC: 0 = NORMAL 1 = TRISTATE
12	TRISTATE	Z_DTB: 0 = NORMAL 1 = TRISTATE
11	TRISTATE	Z_DTA: 0 = NORMAL 1 = TRISTATE
10	TRISTATE	Z_DAP4: 0 = NORMAL 1 = TRISTATE
9	TRISTATE	Z_DAP3: 0 = NORMAL 1 = TRISTATE
8	TRISTATE	Z_DAP2: 0 = NORMAL 1 = TRISTATE
7	TRISTATE	Z_DAP1: 0 = NORMAL 1 = TRISTATE
6	TRISTATE	Z_CSUS: 0 = NORMAL 1 = TRISTATE
5	TRISTATE	Z_CDEV2: 0 = NORMAL 1 = TRISTATE
4	TRISTATE	Z_CDEV1: 0 = NORMAL 1 = TRISTATE
3	TRISTATE	Z_ATD: 0 = NORMAL 1 = TRI TATE

Bit	Reset	Description
2	TRISTATE	Z_ATC: 0 = NORMAL 1 = TRISTATE
1	TRISTATE	Z_ATB: 0 = NORMAL 1 = TRISTATE
0	TRISTATE	Z_ATA: 0 = NORMAL 1 = TRISTATE

11.1.2.2 APB_MISC_PP_TRISTATE_REG_B_0

These registers are used to control the PAD behavior. The TRISTATE registers should be programmed before tri-state pads which are not used for Power consumptions. Z_ indicates the enable for TRISTATING the pads, Enum TRISTATE indicates Outputs are Tristate, and NORMAL indicates they are Not Tristated.

TRI STATES THE Output Pads when the Tristate Bit is enabled. This is used by SW for Power Consumption.

TRI_STATE Configuration Register

Offset: 018h | Read/Write: R/W | Reset: 0b000xx00x11111111111101111111x1110

Bit	Reset	Description
31	NORMAL	Z_DDC: 0 = NORMAL 1 = TRISTATE
30	NORMAL	Z_GMD: 0 = NORMAL 1 = TRISTATE
29	NORMAL	Z_GMB: 0 = NORMAL 1 = TRISTATE
26	NORMAL	Z_KBCC: 0 = NORMAL 1 = TRISTATE
25	NORMAL	Z_ATE: 0 = NORMAL 1 = TRISTATE
23	TRISTATE	Z_UCB: 0 = NORMAL 1 = TRISTATE
22	TRISTATE	Z_UCA: 0 = NORMAL 1 = TRISTATE
21	TRISTATE	Z_UAD: 0 = NORMAL 1 = TRISTATE

Bit	Reset	Description
20	TRISTATE	Z_UAC: 0 = NORMAL 1 = TRISTATE
19	TRISTATE	Z_UAB: 0 = NORMAL 1 = TRISTATE
18	TRISTATE	Z_UAA: 0 = NORMAL 1 = TRISTATE
17	TRISTATE	Z_SPIH: 0 = NORMAL 1 = TRISTATE
16	TRISTATE	Z_SPIG: 0 = NORMAL 1 = TRISTATE
15	TRISTATE	Z_SPIF: 0 = NORMAL 1 = TRISTATE
14	TRISTATE	Z_SPIE: 0 = NORMAL 1 = TRISTATE
13	TRISTATE	Z_SPID: 0 = NORMAL 1 = TRISTATE
12	NORMAL	Z_SPIC: 0 = NORMAL 1 = TRISTATE
11	TRISTATE	Z_SPIB: 0 = NORMAL 1 = TRISTATE
10	TRISTATE	Z_SPIA: 0 = NORMAL 1 = TRISTATE
9	TRISTATE	Z_SPDO: 0 = NORMAL 1 = TRISTATE
8	TRISTATE	Z_SPDI: 0 = NORMAL 1 = TRISTATE
7	TRISTATE	Z_SLXK: 0 = NORMAL 1 = TRISTATE
6	TRISTATE	Z_SLXD: 0 = NORMAL 1 = TRISTATE

Bit	Reset	Description
5	TRISTATE	Z_SLXC: 0 = NORMAL 1 = TRISTATE
3	TRISTATE	Z_SLXA: 0 = NORMAL 1 = TRISTATE
2	TRISTATE	Z_SDD: 0 = NORMAL 1 = TRISTATE
1	TRISTATE	Z_SDC: 0 = NORMAL 1 = TRISTATE
0	NORMAL	Z_GME: 0 = NORMAL 1 = TRISTATE

11.1.2.3 APB_MISC_PP_TRISTATE_REG_C_0

These registers are used to control the PAD behavior. The TRISTATE registers should be programmed before tri-state pads which are not used for Power consumptions. Z_ indicates the enable for TRISTATING the pads, Enum TRISTATE indicates Outputs are Tristate, and NORMAL indicates they are Not Tristated.

TRI STATES THE Output Pads when the Tristate Bit is enabled. This is used by SW for Power Consumption.

TRI_STATE Configuration Register

Offset: 01ch | Read/Write: R/W | Reset: 0b11111111111111111111111111111111

Bit	Reset	Description
31	TRISTATE	Z_LCSN: 0 = NORMAL 1 = TRISTATE
30	TRISTATE	Z_LDC: 0 = NORMAL 1 = TRISTATE
29	TRISTATE	Z_LSCK: 0 = NORMAL 1 = TRISTATE
28	TRISTATE	Z_LSC1: 0 = NORMAL 1 = TRISTATE
27	TRISTATE	Z_LSC0: 0 = NORMAL 1 = TRISTATE
26	TRISTATE	Z_LVS: 0 = NORMAL 1 = TRISTATE

Bit	Reset	Description
25	TRISTATE	Z_LM1: 0 = NORMAL 1 = TRISTATE
24	TRISTATE	Z_LM0: 0 = NORMAL 1 = TRISTATE
23	TRISTATE	Z_HDINT: 0 = NORMAL 1 = TRISTATE
22	TRISTATE	Z_LVP1: 0 = NORMAL 1 = TRISTATE
21	TRISTATE	Z_LVP0: 0 = NORMAL 1 = TRISTATE
20	TRISTATE	Z_LHP2: 0 = NORMAL 1 = TRISTATE
19	TRISTATE	Z_LHP1: 0 = NORMAL 1 = TRISTATE
18	TRISTATE	Z_LHP0: 0 = NORMAL 1 = TRISTATE
17	TRISTATE	Z_LD17: 0 = NORMAL 1 = TRISTATE
16	TRISTATE	Z_LD16: 0 = NORMAL 1 = TRISTATE
15	TRISTATE	Z_LD15: 0 = NORMAL 1 = TRISTATE
14	TRISTATE	Z_LD14: 0 = NORMAL 1 = TRISTATE
13	TRISTATE	Z_LD13: 0 = NORMAL 1 = TRISTATE
12	TRISTATE	Z_LD12: 0 = NORMAL 1 = TRISTATE
11	TRISTATE	Z_LD11: 0 = NORMAL 1 = TRISTATE

Bit	Reset	Description
10	TRISTATE	Z_LD10: 0 = NORMAL 1 = TRISTATE
9	TRISTATE	Z_LD9: 0 = NORMAL 1 = TRISTATE
8	TRISTATE	Z_LD8: 0 = NORMAL 1 = TRISTATE
7	TRISTATE	Z_LD7: 0 = NORMAL 1 = TRISTATE
6	TRISTATE	Z_LD6: 0 = NORMAL 1 = TRISTATE
5	TRISTATE	Z_LD5: 0 = NORMAL 1 = TRISTATE
4	TRISTATE	Z_LD4: 0 = NORMAL 1 = TRISTATE
3	TRISTATE	Z_LD3: 0 = NORMAL 1 = TRISTATE
2	TRISTATE	Z_LD2: 0 = NORMAL 1 = TRISTATE
1	TRISTATE	Z_LD1: 0 = NORMAL 1 = TRISTATE
0	TRISTATE	Z_LD0: 0 = NORMAL 1 = TRISTATE

11.1.2.4 APB_MISC_PP_TRISTATE_REG_D_0

These registers are used to control the PAD behavior. The TRISTATE registers should be programmed before tri-state pads which are not used for Power consumptions.

Z_ indicates the enable for TRISTATING the pads, Enum TRISTATE indicates Outputs are Tristate, and NORMAL indicates they are Not Tristated.

TRI STATES THE Output Pads when the Tristate Bit is enabled. This is used by SW for Power Consumption.

TRI_STATE Configuration Register

Offset: 020h | Read/Write: R/W | Reset: 0b111100x111111111

Bit	Reset	Description
15	TRISTATE	Z_SDB: 0 = NORMAL 1 = TRISTATE
14	TRISTATE	Z_CRTP: 0 = NORMAL 1 = TRISTATE
13	TRISTATE	Z_UDA: 0 = NORMAL 1 = TRISTATE
12	TRISTATE	Z_DTF: 0 = NORMAL 1 = TRISTATE
11	NORMAL	Z_GPU7: 0 = NORMAL 1 = TRISTATE
10	NORMAL	Z_KBCD: 0 = NORMAL 1 = TRISTATE
8	TRISTATE	Z_LPP: 0 = NORMAL 1 = TRISTATE
7	TRISTATE	Z_LHS: 0 = NORMAL 1 = TRISTATE
6	TRISTATE	Z_LDI: 0 = NORMAL 1 = TRISTATE
5	TRISTATE	Z_LPW2: 0 = NORMAL 1 = TRISTATE
4	TRISTATE	Z_LPW1: 0 = NORMAL 1 = TRISTATE
3	TRISTATE	Z_LPW0: 0 = NORMAL 1 = TRISTATE
2	TRISTATE	Z_LSDI: 0 = NORMAL 1 = TRISTATE
1	TRISTATE	Z_LSDA: 0 = NORMAL 1 = TRISTATE
0	TRISTATE	Z_LSPI: 0 = NORMAL 1 = TRISTATE

11.1.3 JTAG Configuration Register

This register controls the basic configuration of the JTAG interface.

11.1.3.1 APB_MISC_PP_CONFIG_CTL_0

Configuration Control Register

Offset: 024h | Read/Write: R/W | Reset: 0b01

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

11.1.4 USB PHY Configuration Registers

11.1.4.1 APB_MISC_PP_MISC_USB_OTG_0

This register is used to control suspend/resume status of USB PHY, and VBUS and ID sensor status and control. Some of the bits in this register also appear in the registers USB_PHY_VBUS_SENSORS and USB_PHY_VBUS_WAKEUP_ID.

Miscellaneous Control Register for OTG Controller

Offset: 028h | Read/Write: R/W | Reset: 0b00xxxxxxx000000000010000x00x0xx0

Bit	R/W	Reset	Description
31	RW	0x0	WAKE_ON_DISCON_EN: Wake on Disconnect Enable (device mode). When enabled (1), USB PHY will wake up from suspend on a disconnect event. 0 = DISABLE 1 = ENABLE
30	RW	0x0	WAKE_ON_CNNT_EN: Wake on Connect Enable (device mode) When enabled (1), USB PHY will wake up from suspend on a connect event. 0 = DISABLE 1 = ENABLE
25	RO	X	B_SESS_END_STS2: B_SESS_END status from USB PHY. This field is the same as the field B_SESS_END_STS in register USB_PHY_VBUS_SENSORS. 0 = UNSET 1 = SET
24	RO	X	A_VBUS_VLD_STS2: A_VBUS_VLD status from USB PHY. This field is the same as the field A_VBUS_VLD_STS in register USB_PHY_VBUS_SENSORS. 0 = UNSET 1 = SET
23	RO	X	A_SESS_VLD_STS2: A_SESS_VLD status from USB PHY. This field is the same as the field A_SESS_VLD_STS in register USB_PHY_VBUS_SENSORS. 0 = UNSET 1 = SET
22	RW	0x0	SW_B_SESS_END: Software_B_SESS_END status. Software should write the appropriate value (1/0) to set/unset the B_SESS_END status. This field is the same as the field B_SESS_END_SW_VALUE in register USB_PHY_VBUS_SENSORS. 0 = UNSET 1 = SET

Bit	R/W	Reset	Description
21	RW	0x0	SW_B_SESS_END_EN: Enable Software Controlled B_SESS_END. Software sets this bit to drive the value in SW_B_SESS_END to the USB controller This field is the same as the field B_SESS_END_SW_EN in register USB_PHY_VBUS_SENSORS. 0 = DISABLE 1 = ENABLE
20	RW	0x0	SW_A_VBUS_VLD: Software_A_VBUS_VLD status. Software should write the appropriate value (1/0) to set/unset the A_VBUS_VLD status. This field is the same as the field A_VBUS_VLD_SW_VALUE in register USB_PHY_VBUS_SENSORS. 0 = UNSET 1 = SET
19	RW	0x0	SW_A_VBUS_VLD_EN: Enable Software Controlled A_VBUS_VLD. Software sets this bit to drive the value in SW_A_VBUS_VLD to the USB controller. This field is the same as the field A_VBUS_VLD_SW_EN in register USB_PHY_VBUS_SENSORS. 0 = DISABLE 1 = ENABLE
18	RW	0x0	SW_A_SESS_VLD: Software_A_SESS_VLD status. Software should write the appropriate value (1/0) to set/unset the A_SESS_VLD status. This field is the same as the field A_SESS_VLD_SW_VALUE in register USB_PHY_VBUS_SENSORS. 0 = UNSET 1 = SET
17	RW	0x0	SW_A_SESS_VLD_EN: Enable Software Controlled A_SESS_VLD. Software sets this bit to drive the value in SW_A_SESS_VLD to the USB controller. This field is the same as the field A_SESS_VLD_SW_EN in register USB_PHY_VBUS_SENSORS. 0 = DISABLE 1 = ENABLE
16	RW	0x0	SUSP_SET: Suspend Set Software must write a 1 to this bit to put the USB PHY in suspend mode. Software should do this only after making sure that the USB is indeed in suspend mode. Setting this bit will stop the PHY clock. Software should write a 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
15	RW	0x0	VBUS_CHG_INT_EN: VBUS Change Interrupt Enable If set, an interrupt will be generated whenever A_SESS_VLD changes value. Software can read the value of A_SESS_VLD from A_SESS_VLD_STS2 bit. This field is the same as the field A_SESS_VLD_INT_EN in register USB_PHY_VBUS_SENSORS. 0 = DISABLE 1 = ENABLE
14	RW	0x0	SW_OTG_ID: Software controlled OTG_ID. If SW_OTG_ID_EN = 1, then software needs to monitor actual OTG_ID bit used as a GPIO and based on the value of OTG_ID, it can set this bit. This field is the same as the field ID_SW_VALUE in register USB_PHY_VBUS_WAKEUP_ID. 0 = UNSET 1 = SET
13	RW	0x0	RSVD13: Reserved
12	RW	0x1	ID_PU: ID pullup enable. This field controls the internal pull-up to OTG_ID pin. Software should set this to 1 if using internal OTG_ID. If software is using a GPIO for OTG_ID, then it can write this to 0. 0 = DISABLE 1 = ENABLE
11	RW	0x0	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

Bit	R/W	Reset	Description
			guarantee proper operation. 0 = UNSET 1 = SET
10	RW	0x0	RSM_VBUS_CHG: Wake/resume on VBUS change detect. If enabled, the USB PHY will wake up whenever a change in A_SESS_VLD is detected. This should be set only when USB PHY is already suspended. 0 = DISABLE 1 = ENABLE
9	RW	0x0	RSM_IE: Resume/Clock valid interrupt enable. If this bit is enabled, interrupt is generated whenever PHY clock becomes valid. 0 = DISABLE 1 = ENABLE
8	RW	0x0	WK_RSM_EN: Wake on resume enable. If this bit is enabled, USB PHY will wake up from suspend whenever resume/reset signaling is detected on USB. 0 = DISABLE 1 = ENABLE
7	RO	X	PCLKVLD: PHY clock valid status. This bit is set whenever PHY clock becomes valid. It is cleared whenever PHY clock stops. 0 = UNSET 1 = SET
6	RO	0x0	VBUS_CHG_DET: VBUS change detect. This bit is set whenever a change in A_SESS_VLD is detected. Software can read the status of A_SESS_VLD from A_SESS_VLD_STS2 bit. This field is the same as the field A_SESS_VLD_CHG_DET in register USB_PHY_VBUS_SENSORS. 0 = UNSET 1 = SET
5	RW	0x0	LPBK_EN: Loopback enable Not for normal software use 0 = DISABLE 1 = ENABLE
4	RO	X	OTG_ID: Real OTG_ID status from the USB PHY. This field is the same as the field ID_STS in register USB_PHY_VBUS_WAKEUP_ID. 0 = UNSET 1 = SET
3	RW	0x0	SW_OTG_ID_EN: Enable Software Controlled OTG_ID If using a GPIO for OTG_ID signal, then software can set this to 1 and write the value from the GPIO to the SW_OTG_ID bit in this register. This field is the same as the field ID_SW_EN in register USB_PHY_VBUS_WAKEUP_ID. 0 = DISABLE 1 = ENABLE
2	RO	X	SUSPENDED: USB PHY suspend status This bit is set to 1 whenever USB is suspended and the PHY clock isn't available. NOTE: Software should not access any registers in USB controller when this bit is set. 0 = UNSET 1 = SET
1	RO	X	StaticGpi: Static General purpose input coming from ID pin 0 = UNSET 1 = SET
0	RW	0x0	SUS_POL: Polarity of the suspend signal going to USB PHY Software should not change this. 0 = ACTIVE_LOW 1 = ACTIVE_HIGH

11.1.4.2 APB_MISC_PP_USB_PHY_PARAM_0

This register controls the parameters for USB PHY.

USB PHY Parameters

Offset: 064h | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
4:3	0x0	VS_CTL: Vbus_sense control Controls which VBUS sensor input is driven to the controller. 00: Use VBUS_WAKEUP. 01: Use (A_SESS_VLD || B_SESS_VLD) output from the PHY if the PHY clock is available. Otherwise, use VBUS_WAKEUP. 10: Use (A_SESS_VLD || B_SESS_VLD) output from the PHY 11: Use A_SESS_VLD output from the PHY 0 = VBUS_WAKEUP 1 = AB_SESS_VLD_OR_VBUS_WAKEUP 2 = AB_SESS_VLD 3 = A_SESS_VLD

11.1.4.3 APB_MISC_PP_USB_PHY_VBUS_SENSORS_0

This register controls the OTG VBUS sensors in the USB PHY.

The following 4 VBUS sensors are present:

- A_VBUS_VLD
- A_SESS_VLD
- B_SESS_VLD
- B_SESS_END

The debounced status of each sensor can be read from _STS. The field _CHG_DET is set to 1 whenever a change is detected in the value. If _INT_EN is set, then an interrupt is generated to the processor. This interrupt can be routed to CPU/COP by appropriately writing the USBD bits in the interrupt controller registers.

In case Software wants to override the value for ‘a’, it can set _SW_EN to 1, and set _SW_VALUE to 1 or 0 as per the requirement.

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 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.
The source for A_SESS_VLD sensor can be programmed according to the setting of VS_CTL in register USB_PHY_PARAM

Offset: 070h | Read/Write: R/W | Reset: 0b000x00xx000x00xx000x00xx000x00

Bit	R/W	Reset	Description
29	RW	0x0	A_VBUS_VLD_DEB_SEL_B: A_VBUS_VLD debounce A/B select. Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
28	RW	0x0	A_VBUS_VLD_SW_VALUE: A_VBUS_VLD software value. Software should write the appropriate value (1/0) to set/unset the A_VBUS_VLD status. This is only valid when A_VBUS_VLD_SW_EN is set.

Bit	R/W	Reset	Description
			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
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
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
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

11.1.4.4 APB_MISC_PP_USB_PHY_VBUS_WAKEUP_ID_0

This register controls the battery charger (VDAT_DET), VBUS_WAKEUP and ID sensors.

The following sensors are present in this register:

- VBUS_WAKEUP
- ID
- VDAT_DET

The debounced status of each sensor can be read from _STS. The field _CHG_DET is set to 1 whenever a change is detected in the value of . If _INT_EN is set, then an interrupt is generated to the processor. This interrupt can be routed to CPU/COP by appropriately writing the USBD bits in the interrupt controller registers.

In case Software wants to override the value for a , it can set _SW_EN to 1, and set _SW_VALUE to 1 or 0 as per the requirement.

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 approrpriate value (SEL_A or SEL_B).

Note: Do not set either UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B to 0x0. If not using one of the debouncers, keep it at the default value of 0xFFFF.

There is no debouncer for VDAT_DET.

USB PHY VBUS wakeup and ID control register

Offset: 074h | Read/Write: R/W | Reset: 0b000110xx000x00xx000x00xx000x00

Bit	R/W	Reset	Description
29:24	RW	0x6	IP_DELAY_TX2TX_HS: HS Tx to Tx inter-packet delay counter. Software should not change this.
21	RW	0x0	VDAT_DET_DEB_SEL_B: VDAT_DET 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	VDAT_DET_SW_VALUE: VDAT_DET software value. Software should write the appropriate value (1/0) to set/unset the VDAT_DET status. This is only valid when VDAT_DET_SW_EN is set. 0 = UNSET 1 = SET
19	RW	0x0	VDAT_DET_SW_EN: VDAT_DET software enable. Enable Software Controlled VDAT_DET. Software sets this bit to drive the value in VDAT_DET_SW_VALUE to the USB controller 0 = DISABLE 1 = ENABLE
18	RO	X	VDAT_DET_STS: VDAT_DET status. This is set to 1 whenever VDAT_DET sensor output is 1. 0 = UNSET 1 = SET
17	RO	0x0	VDAT_DET_CHG_DET: VDAT_DET change detect. This field is set by hardware whenever a change is detected in the value of VDAT_DET. software writes a 1 to clear it

Bit	R/W	Reset	Description
			0 = UNSET 1 = SET
16	RW	0x0	VDAT_DET_INT_EN: VDAT_DET interrupt enable. If this field is set to 1, an interrupt is generated whenever VDAT_DET_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
13	RW	0x0	VBUS_WAKEUP_DEB_SEL_B: VBUS_WAKEUP debounce A/B select. Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
12	RW	0x0	VBUS_WAKEUP_SW_VALUE: VBUS_WAKEUP software value. Software should write the appropriate value (1/0) to set/unset the VBUS_WAKEUP status. This is only valid when VBUS_WAKEUP_SW_EN is set. 0 = UNSET 1 = SET
11	RW	0x0	VBUS_WAKEUP_SW_EN: VBUS_WAKEUP software enable. Enable Software Controlled VBUS_WAKEUP. Software sets this bit to drive the value in VBUS_WAKEUP_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
10	RO	X	VBUS_WAKEUP_STS: VBUS_WAKEUP status. This is set to 1 whenever VBUS_WAKEUP sensor output is 1. 0 = UNSET 1 = SET
9	RO	0x0	VBUS_WAKEUP_CHG_DET: VBUS_WAKEUP change detect. This field is set by hardware whenever a change is detected in the value of VBUS_WAKEUP. software writes a 1 to clear it 0 = UNSET 1 = SET
8	RW	0x0	VBUS_WAKEUP_INT_EN: VBUS_WAKEUP interrupt enable If this field is set to 1, an interrupt is generated whenever VBUS_WAKEUP_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
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

Bit	R/W	Reset	Description
			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

11.1.4.5 APB_MISC_PP_USB_PHY_ALT_VBUS_STS_0

USB PHY Alternate VBUS status register

Offset: 078h | Read/Write: RO | Reset: 0bxxxxxxx

Bit	Reset	Description
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

11.1.5 Pad Pin-Mux Control Registers

The pin mux registers given below are used to select the digital interfaces. These registers need to be programmed during the initial stages of the pin group.

Each pin group can support up to 4 options, specified below. The enumerated values represent the settings PRIMARY, ALT-1, ALT-2, or ALT-3 for each pin group in that order.

RSVD denotes the Pin option is not present.

For eg:- SPI2 denotes the pin option for SPI Second controller. Similarly, UARTA denotes pin option for first UART controller

For the Pin Mux Selection register, a particular interface is selected based on the select signals. Refer to the "*Tegra 2 Series Applications Processor Electrical, Mechanical & Thermal Specifications*" Data Sheet for more explanation.

11.1.5.1 APB_MISC_PP_PIN_MUX_CTL_A_0

PIN Multiplex Control Register A

Offset: 080h | Read/Write: R/W | Reset: 0b000000101010xx100010xx0000000000

Bit	Reset	Description
31:30	0x0	SDIO1_SEL: 0 = SDIO1 1 = RSVD1 2 = UART5 3 = UART1
29:28	0x0	KBCE_SEL: 0 = KBC 1 = NAND 2 = OWR 3 = RSVD2
27:26	0x0	KBCF_SEL: 0 = KBC 1 = NAND 2 = TRACE 3 = MIO
25:24	0x2	ATA_SEL: 0 = IDE 1 = NAND 2 = GMI 3 = RSVD
23:22	0x2	ATC_SEL: 0 = IDE 1 = NAND 2 = GMI 3 = SDIO4
21:20	0x2	ATD_SEL: 0 = IDE 1 = NAND 2 = GMI 3 = SDIO4

Bit	Reset	Description
17:16	0x2	ATB_SEL: 0 = IDE 1 = NAND 2 = GMI 3 = SDIO4
15:14	0x0	RM_SEL: 0 = I2C 1 = RSVD1 2 = RSVD2 3 = RSVD3
13:12	0x0	ATE_SEL: 0 = IDE 1 = NAND 2 = GMI 3 = RSVD
9:8	0x0	UDA_SEL: 0 = SPI1 1 = RSVD 2 = UART4 3 = ULPI
7:6	0x0	UAD_SEL: 0 = IRDA 1 = SPDIF 2 = UART1 3 = SPI4
5:4	0x0	UAC_SEL: 0 = OWR 1 = RSVD2 2 = RSVD3 3 = RSVD4
3:2	0x0	UAB_SEL: 0 = SPI2 1 = MIPI_HS 2 = UART1 3 = ULPI
1:0	0x0	UAA_SEL: 0 = SPI3 1 = MIPI_HS 2 = UART1 3 = ULPI

11.1.5.2 APB_MISC_PP_PIN_MUX_CTL_B_0

PIN Multiplex Control Register B

Offset: 084h | Read/Write: R/W | Reset: 0b000000xx00001010000101000000

Bit	Reset	Description

Bit	Reset	Description
31:30	0x0	DTE_SEL: 0 = RSVD1 1 = RSVD2 2 = VI 3 = SPI1
29:28	0x0	DTD_SEL: 0 = RSVD1 1 = SDIO2 2 = VI 3 = RSVD4
27:26	0x0	DTC_SEL: 0 = RSVD1 1 = RSVD2 2 = VI 3 = RSVD4
23:22	0x0	DTB_SEL: 0 = RSVD1 1 = RSVD2 2 = VI 3 = SPI1
21:20	0x0	DTA_SEL: 0 = RSVD1 1 = SDIO2 2 = VI 3 = RSVD4
19:18	0x2	UCB_SEL: 0 = UART3 1 = PWM 2 = GMI 3 = RSVD4
17:16	0x2	UCA_SEL: 0 = UART3 1 = RSVD2 2 = GMI 3 = RSVD4
15:14	0x0	SLXK_SEL: 0 = PCIE 1 = SPI4 2 = SDIO3 3 = SPI2
13:12	0x1	SLXD_SEL: 0 = SPDIF 1 = SPI4 2 = SDIO3 3 = SPI2

Bit	Reset	Description
11:10	0x1	SLXC_SEL: 0 = SPDIF 1 = SPI4 2 = SDIO3 3 = SPI2
9:8	0x0	OWC_SEL: 0 = OWR 1 = RSVD1 2 = RSVD2 3 = RSVD3
7:6	0x0	SLXA_SEL: 0 = PCIE 1 = SPI4 2 = SDIO3 3 = SPI2
5:4	0x0	HDINT_SEL: 0 = HDMI 1 = RSVD2 2 = RSVD3 3 = RSVD4
3:2	0x2	GMC_SEL: 0 = UART4 1 = SPI4 2 = GMI 3 = SFLASH
1:0	0X2	GMA_SEL: 0 = UART5 1 = SPI3 2 = GMI 3 = SDMMC4

11.1.5.3 APB_MISC_PP_PIN_MUX_CTL_C_0

PIN Multiplex Control Register C

Offset: 088h | Read/Write: R/W | Reset: 0b10101000110010100000000000000000

Bit	Reset	Description
31:30	0x2	GMD_SEL: 0 = RSVD1 1 = NAND 2 = GMI 3 = SFLASH
29:28	0x2	GMB_SEL: 0 = IDE 1 = NAND 2 = GMI 3 = GMI_INT

Bit	Reset	Description
27:26	0x2	DAP4_SEL: 0 = DAP4 1 = RSVD2 2 = GMI 3 = RSVD4
25:24	0x0	DAP3_SEL: 0 = DAP3 1 = RSVD2 2 = RSVD3 3 = RSVD4
23:22	0x0	DAP2_SEL: 0 = DAP2 1= TWC 2 = RSVD3 3 = GMI
21:20	0x0	DAP1_SEL: 0 = DAP1 1 = RSVD2 2 = GMI 3 = SDIO2
19:18	0x2	IRRX_SEL: 0 = UART1 1 = UART2 2 = GMI 3 = SPI4
17:16	0x2	IRTX_SEL: 0 = UART1 1 = UART2 2 = GMI 3 = SPI4
15:14	0x0	KBCC_SEL: 0 = KBC 1 = NAND 2 = TRACE 3 = EMC_TEST1_DLL
13:12	0x0	KBCB_SEL: 0 = KBC 1 = NAND 2 = SDIO2 3 = MIO
11:10	0x0	KBCA_SEL: 0 = KBC 1 = NAND 2 = SDIO2 3 = EMC_TEST1_DLL

Bit	Reset	Description
9:8	0x0	I2CP_SEL: 0 = I2C 1 = RSVD2 2 = RSVD3 3 = RSVD4
7:6	0x0	CSUS_SEL: 0 = PLLC_OUT1 1 = PLLP_OUT2 2 = PLLP_OUT3 3 = VI_SENSOR_CLK
5:4	0x0	CDEV2_SEL: 0 = OSC 1 = AHB_CLK 2 = APB_CLK 3 = PLLP_OUT4
3:2	0x0	CDEV1_SEL: 0 = OSC 1 = PLLA_OUT 2 = PLLM_OUT1 3 = AUDIO_SYNC
1:0	0x0	DDC_SEL: 0 = I2C2 1 = RSVD1 2 = RSVD2 3 = RSVD3

11.1.5.4 APB_MISC_PP_PIN_MUX_CTL_D_0

PIN Multiplex Control Register D

Offset: 08ch | Read/Write: R/W | Reset: 0b11111111110000000000000000010

Bit	Reset	Description
31:30	0x3	SPIA_SEL: 0 = SPI1 1 = SPI2 2 = SPI3 3 = GMI
29:28	0x3	SPIB_SEL: 0 = SPI1 1 = SPI2 2 = SPI3 3 = GMI
27:26	0x3	SPIC_SEL: 0 = SPI1 1 = SPI2 2 = SPI3 3 = GMI

Bit	Reset	Description
25:24	0x3	SPID_SEL: 0 = SPI2 1 = SPI1 2 = SPI2_ALT 3 = GMI
23:22	0x3	SPIE_SEL: 0 = SPI2 1 = SPI1 2 = SPI2_ALT 3 = GMI
21:20	0x0	SPIF_SEL: 0 = SPI3 1 = SPI1 2 = SPI2 3 = RSVD
19:18	0x0	SPIG_SEL: 0 = SPI3 1 = SPI2 2 = SPI2_ALT 3 = I2C
17:16	0x0	SPIH_SEL: 0 = SPI3 1 = SPI2 2 = SPI2_ALT 3 = I2C
15:14	0x0	SDD_SEL: 0 = UART1 1 = PWM 2 = SDIO3 3 = SPI3
13:12	0x0	SDC_SEL: 0 = PWM 1 = TWC 2 = SDIO3 3 = SPI3
11:10	0x0	SDB_SEL: 0 = UART1 1 = PWM 2 = SDIO3 3 = SPI2
9:8	0x0	SPDI_SEL: 0 = SPDIF 1 = RSVD 2 = I2C 3 = SDIO2

Bit	Reset	Description
7:6	0x0	SPDO_SEL: 0 = SPDIF 1 = RSVD 2 = I2C 3 = SDIO2
5:4	0x0	GPU_SEL: 0 = PWM 1 = UART1 2 = GMI 3 = RSVD4
3:2	0x0	GPV_SEL: 0 = PCIE 1 = RSVD2 2 = RSVD3 3 = RSVD4
1:0	0x2	GME_SEL: 0 = RSVD1 1 = DAP5 2 = GMI 3 = SDIO4

11.1.5.5 APB_MISC_PP_PIN_MUX_CTL_E_0

PIN Multiplex Control Register E

Offset: 090h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:30	0x0	LVP0_SEL: 0 = LCD1 1 = LCD2 2 = RSVD3 3 = RSVD
29:28	0x0	LM1_SEL: 0 = LCD1 1 = LCD2 2 = RSVD3 3 = CRT
27:26	0x0	LM0_SEL: 0 = LCD1 1 = LCD2 2 = SPI3 3 = RSVD
25:24	0x0	LVS_SEL: 0 = LCD1 1 = LCD2 2 = XIO 3 = RSVD

Bit	Reset	Description
23:22	0x0	LHS_SEL: 0 = LCD1 1 = LCD2 2 = XIO 3 = RSVD
21:20	0x0	LSC1_SEL: 0 = LCD1 1 = LCD2 2 = SPI3 3 = HDMI
19:18	0x0	LSC0_SEL: 0 = LCD1 1 = LCD2 2 = XIO 3 = RSVD
17:16	0x0	LSCK_SEL: 0 = LCD1 1 = LCD2 2 = SPI3 3 = HDMI
15:14	0x0	LDC_SEL: 0 = LCD1 1 = LCD2 2 = RSVD 3 = RSVD4
13:12	0x0	LCSN_SEL: 0 = LCD1 1 = LCD2 2 = SPI3 3 = RSVD4
11:10	0x0	LSPI_SEL: 0 = LCD1 1 = LCD2 2 = XIO 3 = HDMI
9:8	0x0	LSDA_SEL: 0 = LCD1 1 = LCD2 2 = SPI3 3 = HDMI
7:6	0x0	LSDI_SEL: 0 = LCD1 1 = LCD2 2 = SPI3 3 = RSVD

Bit	Reset	Description
5:4	0x0	LPW2_SEL: 0 = LCD1 1 = LCD2 2 = SPI3 3 = HDMI
3:2	0x0	LPW1_SEL: 0 = LCD1 1 = LCD2 2 = RSVD 3 = RSVD4
1:0	0x0	LPW0_SEL: 0 = LCD1 1 = LCD2 2 = SPI3 3 = HDMI

11.1.5.6 APB_MISC_PP_PIN_MUX_CTL_F_0

PIN Multiplex Control Register F

Offset: 094h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:30	0x0	LD15_SEL: 0 = LCD1 1 = LCD2 2 = XIO 3 = RSVD
29:28	0x0	LD14_SEL: 0 = LCD1 1 = LCD2 2 = XIO 3 = RSVD
27:26	0x0	LD13_SEL: 0 = LCD1 1 = LCD2 2 = XIO 3 = RSVD
25:24	0x0	LD12_SEL: 0 = LCD1 1 = LCD2 2 = XIO 3 = RSVD
23:22	0x0	LD11_SEL: 0 = LCD1 1 = LCD2 2 = XIO 3 = RSVD

Bit	Reset	Description
21:20	0x0	LD10_SEL: 0 = LCD1 1 = LCD2 2 = XIO 3 = RSVD
19:18	0x0	LD9_SEL: 0 = LCD1 1 = LCD2 2 = XIO 3 = RSVD
17:16	0x0	LD8_SEL: 0 = LCD1 1 = LCD2 2 = XIO 3 = RSVD
15:14	0x0	LD7_SEL: 0 = LCD1 1 = LCD2 2 = XIO 3 = RSVD
13:12	0x0	LD6_SEL: 0 = LCD1 1 = LCD2 2 = XIO 3 = RSVD
11:10	0x0	LD5_SEL: 0 = LCD1 1 = LCD2 2 = XIO 3 = RSVD
9:8	0x0	LD4_SEL: 0 = LCD1 1 = LCD2 2 = XIO 3 = RSVD
7:6	0x0	LD3_SEL: 0 = LCD1 1 = LCD2 2 = XIO 3 = RSVD
5:4	0x0	LD2_SEL: 0 = LCD1 1 = LCD2 2 = XIO 3 = RSVD

Bit	Reset	Description
3:2	0x0	LD1_SEL: 0 = LCD1 1 = LCD2 2 = XIO 3 = RSVD
1:0	0x0	LD0_SEL: 0 = LCD1 1 = LCD2 2 = XIO 3 = RSVD

11.1.5.7 APB_MISC_PP_PIN_MUX_CTL_G_0

PIN Multiplex Control Register G

Offset: 098h | Read/Write: R/W | Reset: 0b000000xx0000000000xx000000000000

Bit	Reset	Description
31:30	0x0	DTF_SEL: 0 = I2C3 1 = RSVD2 2 = VI 3 = RSVD4
29:28	0x0	GPU7_SEL: 0 = RTCK 1 = RSVD1 2 = RSVD2 3 = RSVD3
27:26	0x0	KBCD_SEL: 0 = KBC 1 = NAND 2 = SDIO2 3 = MIO
23:22	0x0	PTA_SEL: 0 = I2C2 1 = HDMI 2 = GMI 3 = RSVD3
21:20	0x0	CRTP_SEL: 0 = CRT 1 = RSVD2 2 = RSVD3 3 = RSVD4
19:18	0x0	PMC_SEL: 0 = PWR_ON 1 = PWR_INTR

Bit	Reset	Description
17:16	0x0	LDI_SEL: 0 = LCD1 1 = LCD2 2 = RSVD3 3 = RSVD
15:14	0x0	LPP_SEL: 0 = LCD1 1 = LCD2 2 = RSVD3 3 = RSVD
11:10	0x0	LHP0_SEL: 0 = LCD1 1 = LCD2 2 = RSVD3 3 = RSVD4
9:8	0x0	LVP1_SEL: 0 = LCD1 1 = LCD2 2 = RSVD3 3 = RSVD4
7:6	0x0	LHP2_SEL: 0 = LCD1 1 = LCD2 2 = RSVD3 3 = RSVD4
5:4	0x0	LHP1_SEL: 0 = LCD1 1 = LCD2 2 = RSVD3 3 = RSVD
3:2	0x0	LD17_SEL: 0 = LCD1 1 = LCD2 2 = RSVD 3 = RSVD
1:0	0x0	LD16_SEL: 0 = LCD1 1 = LCD2 2 = XIO 3 = RSVD

11.1.5.8 APB_MISC_PP_PIN_MUX_CTL_H_0

PIN Multiplex Control Register H. This register is used for DAP-DAC selections.

Offset: 09ch | Read/Write: R/W | Reset: 0b00000000000000000000000000

Bit	Reset	Description
21	0x0	DAP4M_SEL: 0 = DAP4_SLAVE 1 = DAP4_MASTER

Bit	Reset	Description
20	0x0	DAP3M_SEL: 0 = DAP3_SLAVE 1 = DAP3_MASTER
19	0x0	DAP2M_SEL: 0 = DAP2_SLAVE 1 = DAP2_MASTER
18	0x0	DAP1M_SEL: 0 = DAP1_SLAVE 1 = DAP1_MASTER
17:16	0x0	DAC3_SEL: 0 = DAP1 1 = DAP2 2 = DAP3 3 = DAP4
15:14	0x0	DAC2_SEL: 0 = DAP1 1 = DAP2 2 = DAP3 3 = DAP4
13:12	0x0	DAC1_SEL: 0 = DAP1 1 = DAP2 2 = DAP3 3 = DAP4
11:9	0x0	DAP4_CNTRL_MUX_SELECT: 0 = DAC1 1 = DAC2 2 = DAC3 3 = RSVD 4 = DAP1 5 = DAP2 6 = DAP3 7 = RSVD2
8:6	0x0	DAP3_CNTRL_MUX_SELECT: 0 = DAC1 1 = DAC2 2 = DAC3 3 = RSVD1 4 = DAP1 5 = DAP2 6 = RSVD2 7 = DAP4
5:3	0x0	DAP2_CNTRL_MUX_SELECT: 0 = DAC1 1 = DAC2 2 = DAC3 3 = RSVD1 4 = DAP1 5 = RSVD2 6 = DAP3 7 = DAP4
2:0	0x0	DAP1_CNTRL_MUX_SELECT: 0 = DAC1 1 = DAC2 2 = DAC3 3 = RSVD1 4 = RSVD2 5 = DAP2 6 = DAP3 7 = DAP4

11.1.6 Pad Pull-up/down Registers

These registers control the pull-up and pull-down enable inputs of the IO pads. These are used to eliminate external components on interfaces that need a pulled up or pulled down when not driven.

Note the following about the pad pull-up and pull-down functions:

- They are weak internal pull-up/pull-downs
- NORMAL setting means pad is neither pulled up (driving weak 1) or pulled down (driving weak 0)
- PULL_DOWN selection means pad is driving weak 0
- PULL_UP selection means pad is driving weak 1

> **WARNING!** Driving internal pull-up when pad has external pull down and vice versa will cause significant increase in power consumption.

11.1.6.1 APB_MISC_PP_PULLUPDOWN_REG_A_0

PULL_UP/PULL_DOWN Control Register

Table 41. Cross Reference of Fields with Ball Names (REG A)

Field Name	Controls Balls
GPV_PU_PD	GPIO_PV[6:4]
DTF_PU_PD	CAM_I2C_SCK, CAM_I2C_SDA
DTE_PU_PD	VI_GP0, VI_GP[6:3]
DTD_PU_PD	VI_PCLK, VI_D[9:2]
DTC_PU_PD	VI_VSYNC, VI_HSYNC
DTB_PU_PD	VID_D[11:10]
DTA_PU_PD	VID_D[1:0]
DAP4_PU_PD	DAP4_FS, DAP4_DIN, DAP4_DOUT, DAP4_SCLK
DAP3_PU_PD	DAP3_FS, DAP3_DIN, DAP3_DOUT, DAP3_SCLK
DAP2_PU_PD	DAP2_FS, DAP2_DIN, DAP2_DOUT, DAP2_SCLK
DAP1_PU_PD	DAP1_FS, DAP1_DIN, DAP1_DOUT, DAP1_SCLK
ATE_PU_PD	GMI_AD15, GMI_AD14, GMI_AD13, GMI_AD12GMI_AD[15:12]
ATD_PU_PD	GMI_AD11, GMI_AD 10, GMI_AD 9, GMI_AD [11:8]
ATC_PU_PD	GMI_IORDY, GMI_WAIT, GMI_CS4ADV_N, GMI_CS3CLK, GMI_CS2CS[4:2]_N, GMI_AD00,AD[7:0], GMI_AD01WR_N, GMI_AD02, GMI_AD03 OE_N
ATB_PU_PD	GMI_CS5_N, GMI_DPD

Field Name	Controls Balls
ATA_PU_PD	GMI_CS6, GMI_CS7CS[7:6]_N, GMI_RST_N

Offset: 0a0h | Read/Write: R/W | Reset: 0b00100001010101010101011010101010

Bit	Reset	Description
31:30	NORMAL	GPV_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
29:28	PULL_UP	DTF_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
27:26	NORMAL	DTE_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
25:24	PULL_DOWN	DTD_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
23:22	PULL_DOWN	DTC_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
21:20	PULL_DOWN	DTB_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
19:18	PULL_DOWN	DTA_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
17:16	PULL_DOWN	DAP4_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
15:14	PULL_DOWN	DAP3_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
13:12	PULL_DOWN	DAP2_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
11:10	PULL_DOWN	DAP1_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
9:8	PULL_UP	ATE_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
7:6	PULL_UP	ATD_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
5:4	PULL_UP	ATC_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
3:2	PULL_UP	ATB_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	PULL_UP	ATA_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

11.1.6.2 APB_MISC_PP_PULLUPDOWN_REG_B_0

PULL_UP/PULL_DOWN Control Register

Table 42. Cross Reference of Fields with Ball Names (REG B)

Field Name	Controls Balls
SLXK_PU_PD	SDIO2_CLK
SLXD_PU_PD	SDIO2_DAT3
SLXC_PU_PD	SDIO2_DAT2
SLXB_PU_PD	SDIO2_DAT1
SLXA_PU_PD	SDIO2_DAT0
GPU_PU_PD	GPIO_PU[6:0]
SPDO_PU_PD	SPDIF_OUT
SPDI_PU_PD	SPDIF_IN
KBCD_PU_PD	KB_ROW[6:4]
KBCC_PU_PD	KB_COL[2:0]
KBCB_PU_PD	KB_ROW[12:7]
KBCA_PU_PD	KB_ROW[2:0]
GPU7_PU_PD	GPIO_PU7
PTA_PU_PD	GEN_I2C_SCL (GEN2_I2C_CLK), GEN_I2C_SDA (SCL, GEN2_I2C_DAT)SDA
I2CP_PU_PD	PWR_I2C_SCL, PWR_I2C_SDA
RM_PU_PD	GEN1_I2C_SDA, GEN1_I2C_SCL

Offset: 0a4h | Read/Write: R/W | Reset: 0b01101010100010000110010110101010

Bit	Reset	Description
31:30	PULL_DOWN	SLXK_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
29:28	PULL_UP	SLXD_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
27:26	PULL_UP	SLXC_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
25:24	PULL_UP	CRTP_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
23:22	PULL_UP	SLXA_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
21:20	NORMAL	GPU_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
19:18	PULL_UP	SPDO_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
17:16	PULL_UP	SPDI_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_U P 3 = RSVD
15:14	PULL_UP	KBCD_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
13:12	PULL_UP	KBCC_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
11:10	PULL_UP	KBCB_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
9:8	PULL_UP	KBCA_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
7:6	PULL_UP	GPU7_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
5:4	PULL_UP	PTA_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
3:2	PULL_UP	I2CP_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	PULL_UP	RM_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

11.1.6.3 APB_MISC_PP_PULLUPDOWN_REG_C_0

PULL_UP/PULL_DOWN Control Register

Table 43. Cross Reference of Fields with Ball Names (REG C)

Field Name	Controls Balls
XM2D_PU_PD	DDR_DQ[31:0]
XM2C_PU_PD	DDR_DQS[2:0]P, DDR_DQS[2:0]N, DDR_DM[3:0], DDR_A[14:0], DDR_BA[2:0], DDR_CKE[1:0], DDR_ODT, DDR_RAS_N, DDR_CAS_N, DDR_WE_N, DDR_CLK, DDR_CLK_N, DDR_QUSE[3:0]
GME_PU_PD	GMI_AD[27:24]
IRRX_PU_PD	UART2_RTS_N
IRTX_PU_PD	UART2_CTS_N
SPIH_PU_PD	SPI2_CS2_N
SPIG_PU_PD	SPI2_CS1_N

Field Name	Controls Balls
SPIF_PU_PD	SPI1_MISO
SPIE_PU_PD	SPI1_SCK, SPI1_CS0_N
SPID_PU_PD	SPI1_MOSI
SPIC_PU_PD	SPI2_SCK, SPI2_CS0_N
SPIB_PU_PD	SPI2_MISO
SPIA_PU_PD	SPI2_MOSI
CDEV2_PU_PD	DAP_MCLK2
CDEV1_PU_PD	DAP_MCLK1

Offset: 0a8h | Read/Write: R/W | Reset: 0b0000xx001010101001100110010101

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
25:24	NORMAL	GME_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
23:22	PULL_UP	IRRX_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
21:20	PULL_UP	IRTX_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
19:18	PULL_UP	SPIH_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
17:16	PULL_UP	SPIG_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
15:14	PULL_DOWN	SPIF_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
13:12	PULL_UP	SPIE_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
11:10	PULL_DOWN	SPID_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
9:8	PULL_UP	SPIC_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
7:6	PULL_DOWN	SPIB_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
5:4	PULL_DOWN	SPIA_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
3:2	PULL_DOWN	CDEV2_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	PULL_DOWN	CDEV1_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

11.1.6.4 APB_MISC_PP_PULLUPDOWN_REG_D_0

PULL_UP/PULL_DOWN Control Register

Table 44. Cross Reference of Fields with Ball Names (REG D)

Field Name	Controls Balls
SDD_PU_PD	SDIO3_CLK,SDIO3_CMD
SDC_PU_PD	SDIO3_DAT[3:0]
SDB_PU_PD	SDIO2_CMD
CSUS_PU_PD	VI_MCLK
LC_PU_PD	LCD_PCLK, LCD_DE, LCD_HSYNC, LCD_VSYNC, LCD_CS1_N, LCD_M1, LCD_DC1, HDMI_INT
LS_PU_PD	LCD_PWR1, LCD_PWR2, LCD_SDIN, LCD_SDOUT, LCD_WR_N, LCD_CS0_N, LCD_DC0, LCD_SCK, LCD_PWR0
LD23_22_PU_PD	LCD_D[23:22]
LD21_20_PU_PD	LCD_D[21:20]
LD19_18_PU_PD	LCD_D[19:18]
LD17_0_PU_PD	LCD_D[17:0]
UCB_PU_PD	UART3_RTS_N, UART3_CTS_N
UCA_PU_PD	UART3_TXD,UART3_RXD
UAD_PU_PD	UART2_TXD,UART2_RXD
UAC_PU_PD	GPIO_PV[3:0]
UAB_PU_PD	UART1_RI_,UART1_DCD_,UART1_DSR_,UART1_DTR_
UAA_PU_PD	UART1_TXD,UART1_RXD,UART1_CTS_,UART1_RTS

Offset: 0ach | Read/Write: R/W | Reset: 0b10100001101001010101101010001010

Bit	Reset	Description
31:30	PULL_UP	SDD_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
29:28	PULL_UP	SDC_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
25:24	PULL_DOWN	CSUS_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
23:22	PULL_UP	LC_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
21:20	PULL_UP	LS_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
19:18	PULL_DOWN	LD23_22_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
17:16	PULL_DOWN	LD21_20_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
15:14	PULL_DOWN	LD19_18_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
13:12	PULL_DOWN	LD17_0_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
11:10	PULL_UP	UCB_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
9:8	PULL_UP	UCA_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
7:6	PULL_UP	UAD_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
5:4	NORMAL	UAC_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
3:2	PULL_UP	UAB_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	PULL_UP	UAA_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

11.1.6.5 APB_MISC_PP_PULLUPDOWN_REG_E_0

Table 45. Cross Reference of Fields with Ball Names (REG E)

Field Name	Controls Balls
OWC_PU_PD	OWR
DDC_PU_PD	DDC_SCL, DDC_SDA
GMD_PU_PD	GMI_CS[1:0]_N
GMC_PU_PD	GMI_AD[19:16]
GMB_PU_PD	GMI_WP_N
GMA_PU_PD	GMI_AD[23:20]
SDIO1_PU_PD	SDIO1_CLK, SDIO1_CMD, SDIO1_DAT[3:0]

Field Name	Controls Balls
UDA_PU_PD	ULPI_CLK, ULPI_DIR, ULPI_NXT, ULPI_STP
CK32_PU_PD	CLK_32K_IN
PMCE_PU_PD	PWR_INT_N
PMCD_PU_PD	CPU_PWR_REQ
PMCC_PU_PD	CORE_PWR_REQ
PMCB_PU_PD	SYS_CLK_REQ
PMCA_PU_PD	CLK_32K_OUT
KBCE_PU_PD	KB_COL7
KBCF_PU_PD	KB_COL[6:2]

Offset: 0b0h | Read/Write: R/W | Reset: 0b10100000000010000000000001010

Bit	Reset	Description
31:30	PULL_UP	OWC_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
29:28	PULL_UP	DDC_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
27:26	NORMAL	GMD_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
25:24	NORMAL	GMC_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
23:22	NORMAL	GMB_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
21:20	NORMAL	GMA_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
19:18	PULL_UP	SDIO1_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
17:16	NORMAL	UDA_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
15:14	NORMAL	CK32_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
13:12	NORMAL	PMCE_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
11:10	NORMAL	PMCD_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
9:8	NORMAL	PMCC_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
7:6	NORMAL	PMCB_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
5:4	NORMAL	PMCA_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
3:2	PULL_UP	KBCE_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	PULL_UP	KBCF_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

11.1.6.6 APB_MISC_PP_MISC_USB_CLK_RST_CTL_0

This register controls the clocks and resets going to USB and USB2 controllers.

USB and USB2 Clock and Reset Register

Offset: 0b4h | Read/Write: R/W | Reset: 0b10xxxxx001

Bit	Reset	Description
9	ENABLE	MISC_USB2_RST: Reset (active high) for USB2 controller 0 = DISABLE 1 = ENABLE
8	DISABLE	MISC_USB_RST: Reset (active high) for USB controller 0 = DISABLE 1 = ENABLE
2	DISABLE	MISC_USB2_CLK_OVR_ON: Clock override for USB2 controller 0 = DISABLE 1 = ENABLE
1	DISABLE	MISC_USB2_CE: enable clocks to USB2 controller 0 = DISABLE 1 = ENABLE
0	ENABLE	MISC_USB_CE: enable clocks to USB controller 0 = DISABLE 1 = ENABLE

11.1.6.7 APB_MISC_GP_MODEREG_0

Offset: 800h | Read/Write: RO | Reset: 0bxxxxxxxxxx

Bit	Reset	Description
9	x	LPDDR_STRAP1: LP-DDR Strap option bit 1.
8	x	LPDDR_STRAP0: LP-DDR Strap option bit 0.
0	x	STANDBY_IE: Standby pad input 1 = STANDBYN is asserted (low voltage), 0 = STANDBYN is desasserted (high voltage) 0 = DEASSERTED 1 = ASSERTED

11.1.6.8 APB_MISC_GP_HIDREV_0

Chip ID Revision Register

Offset: 804h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
19:16	x	MINORREV: Chip ID minor revision (IF MAJORREV==0(Emulation) THEN 0: QT, 1:E388 FPGA)
15:8	x	CHIPID: Chip ID
7:4	x	MAJORREV: Chip ID major revision (0: Emulation, 1-15: Silicon) 0 = EMULATION 1 = A01

Bit	Reset	Description
3:0	x	HIDFAM: Chip ID family register. 7 = NVIDIA Mobile Applications Processor

11.1.6.9 APB_MISC_GP_EMU_REVID_0

EMULATION SCRATCH REGISTERS

Offset: 860h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	NV_EMUL_PATCH	PATCH: USED by emulators to indicate patch #, 0 for silicon.
15:0	NV_EMUL_NETLIST	NETLIST: USED by emulators to indicate netlist #, 0 for silicon.

11.1.6.10 APB_MISC_GP_TRANSACTOR_SCRATCH_0

Offset: 864h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	TRANSACTOR_SCRATCH: used for emulation to determine test results.

11.1.7 Pad Control Registers

The registers described below control pad behavior. Pull-up and pull-down functions are controlled by the dedicated registers above. For each digital pad, the following controls can be used to tune pad performance, and power consumption

- 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 matching timing is important.
- 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 below depicts the function of low power mode:

Table 46. 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 = 50ohm)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:

Table 47. Driver Output Pull-Up drive strength code

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.

11.1.7.1 APB_MISC_GP_AOCFG1PADCTRL_0

Controls the following pins:

- SYS_RESET_
- PWR_I2C_SCL
- PWR_I2C_SDA
- KB_ROW[7:0]

AOCFG1 Pad Control Register

Offset: 868h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_AOCFG1_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_AOCFG1_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_AOCFG1_CAL_DRVUP
16:12	0x12	CFG2TMC_AOCFG1_CAL_DRVDN
5:4	0x3	CFG2TMC_AOCFG1_LPMD: AOCFG1 data pins low power mode select
3	0x0	CFG2TMC_AOCFG1_SCHMT_EN: AOCFG1 data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_AOCFG1_HSM_EN: AOCFG1 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

11.1.7.2 APB_MISC_GP_AOCFG2PADCTRL_0

Controls the following pins:

- KB_ROW[12:8]
- KB_COL[6:0]
- LED_BLINK

- SYS_CLK_REQ
- CORE_PWR_REQ
- CPU_PWR_REQ
- PWR_INT_
- CLK_32K_IN (CK32KHZ)

AOCFG2 Pad Control Register

Offset: 86ch | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_AOCFG2_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_AOCFG2_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_AOCFG2_CAL_DRVUP
16:12	0x12	CFG2TMC_AOCFG2_CAL_DRVDN
5:4	0x3	CFG2TMC_AOCFG2_LPMD: AOCFG2 data pins low power mode select
3	0x0	CFG2TMC_AOCFG2_SCHMT_EN: AOCFG2 data pins Schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_AOCFG2_HSM_EN: AOCFG2 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

11.1.7.3 APB_MISC_GP_ATCFG1PADCTRL_0

Controls the following pins:

- GMI_IORDY
- GMI_AD[15:8]
- GMI_CS7
- GMI_DPD
- GEN_I2C_SCL (GEN2_I2C_CLK)
- GEN_I2C_SDA (GEN2_I2C_DAT)

ATCFG1 Pad Control Register

Offset: 870h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_ATCFG1_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_ATCFG1_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_ATCFG1_CAL_DRVUP
16:12	0x12	CFG2TMC_ATCFG1_CAL_DRVDN
5:4	0x3	CFG2TMC_ATCFG1_LPMD: ATCFG1 data pins low power mode select

Bit	Reset	Description
3	0x0	CFG2TMC_ATCFG1_SCHMT_EN: ATCFG1 data pins Schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_ATCFG1_HSM_EN: ATCFG1 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

11.1.7.4 APB_MISC_GP_ATCFG2PADCTRL_0

Controls the following pins:

- GMI_WAIT
- GMI_ADV_N
- GMI_CLK
- GMI_CS[6:2]
- GMI_AD[7:0]
- GMI_HIOW
- GMI_HIOR
- GMI_RST_N

ATCFG2 Pad Control Register

Offset: 874h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_ATCFG2_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_ATCFG2_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_ATCFG2_CAL_DRVUP
16:12	0x12	CFG2TMC_ATCFG2_CAL_DRVDN
5:4	0x3	CFG2TMC_ATCFG2_LPMD: ATCFG2 data pins low power mode select
3	0x0	CFG2TMC_ATCFG2_SCHMT_EN: ATCFG2 data pins Schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_ATCFG2_HSM_EN: ATCFG2 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

11.1.7.5 APB_MISC_GP_CDEV1CFGPADCTRL_0

Controls:

- DAP_MCLK1

CDEV1CFG Pad Control Register

Offset: 878h | Read/Write: R/W | Reset: 0b1111xxx10110xxx0010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_CDEV1CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_CDEV1CFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_CDEV1CFG_CAL_DRVUP
16:12	0x12	CFG2TMC_CDEV1CFG_CAL_DRVDN
5:4	0x3	CFG2TMC_CDEV1CFG_LPMD: CDEV1CFG data pins low power mode select
3	0x0	CFG2TMC_CDEV1CFG_SCHMT_EN: CDEV1CFG data pins Schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_CDEV1CFG_HSM_EN: CDEV1CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

11.1.7.6 APB_MISC_GP_CDEV2CFGPADCTRL_0

Controls:

- DAP_MCLK2

CDEV2CFG Pad Control Register

Offset: 87ch | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_CDEV2CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_CDEV2CFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_CDEV2CFG_CAL_DRVUP
16:12	0x12	CFG2TMC_CDEV2CFG_CAL_DRVDN
5:4	0x3	CFG2TMC_CDEV2CFG_LPMD: CDEV2CFG data pins low power mode select
3	0x0	CFG2TMC_CDEV2CFG_SCHMT_EN: CDEV2CFG data pins Schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_CDEV2CFG_HSM_EN: CDEV2CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

11.1.7.7 APB_MISC_GP_CSUSCFGPADCTRL_0

Controls:

- VI_MCLK

CSUSCFG Pad Control Register

Offset: 880h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description

Bit	Reset	Description
31:30	0x3	CFG2TMC_CSUSCFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_CSUSCFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_CSUSCFG_CAL_DRVUP
16:12	0x12	CFG2TMC_CSUSCFG_CAL_DRVDN
5:4	0x3	CFG2TMC_CSUSCFG_LPMD: CSUSCFG data pins low power mode select
3	0x0	CFG2TMC_CSUSCFG_SCHMT_EN: CSUSCFG data pins Schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_CSUSCFG_HSM_EN: CSUSCFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

11.1.7.8 APB_MISC_GP_DAP1CFGPADCTRL_0

Controls the following pins:

- DAP1_FS
- DAP1_DIN
- DAP1_DOUT
- DAP1_SCLK
- SPDIF_OUT
- SPDIF_IN

DAP1CFG Pad Control Register

Offset: 884h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_DAP1CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_DAP1CFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_DAP1CFG_CAL_DRVUP
16:12	0x12	CFG2TMC_DAP1CFG_CAL_DRVDN
5:4	0x3	CFG2TMC_DAP1CFG_LPMD: DAP1CFG data pins low power mode select
3	0x0	CFG2TMC_DAP1CFG_SCHMT_EN: DAP1CFG data pins Schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_DAP1CFG_HSM_EN: DAP1CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

11.1.7.9 APB_MISC_GP_DAP2CFGPADCTRL_0

Controls the following pins:

- DAP2_FS
- DAP2_SCLK
- DAP2_DIN
- DAP2_DOUT

DAP2CFG Pad Control Register

Offset: 888h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_DAP2CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_DAP2CFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_DAP2CFG_CAL_DRVUP
16:12	0x12	CFG2TMC_DAP2CFG_CAL_DRVDN
5:4	0x3	CFG2TMC_DAP2CFG_LPMD: DAP2CFG data pins low power mode select
3	0x0	CFG2TMC_DAP2CFG_SCHMT_EN: DAP2CFG data pins Schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_DAP2CFG_HSM_EN: DAP2CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

11.1.7.10 APB_MISC_GP_DAP3CFGPADCTRL_0

Controls the following pins:

- DAP3_FS
- DAP3_DIN
- DAP3_DOUT
- DAP3_SCLK

DAP3CFG Pad Control Register

Offset: 88ch | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_DAP3CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_DAP3CFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_DAP3CFG_CAL_DRVUP
16:12	0x12	CFG2TMC_DAP3CFG_CAL_DRVDN
5:4	0x3	CFG2TMC_DAP3CFG_LPMD: DAP3CFG data pins low power mode select

Bit	Reset	Description
3	0x0	CFG2TMC_DAP3CFG_SCHMT_EN: DAP3CFG data pins Schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_DAP3CFG_HSM_EN: DAP3CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

11.1.7.11 APB_MISC_GP_DAP4CFGPADCTRL_0

Controls the following pins:

- DAP4_FS
- DAP4_DIN
- DAP4_DOUT
- DAP4_SCLK

DAP4CFG Pad Control Register

Offset: 890h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_DAP4CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_DAP4CFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_DAP4CFG_CAL_DRVUP
16:12	0x12	CFG2TMC_DAP4CFG_CAL_DRVDN
5:4	0x3	CFG2TMC_DAP4CFG_LPMD: DAP4CFG data pins low power mode select
3	0x0	CFG2TMC_DAP4CFG_SCHMT_EN: DAP4CFG data pins Schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_DAP4CFG_HSM_EN: DAP4CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

11.1.7.12 APB_MISC_GP_DBGCFGPADCTRL_0

Controls the following pins:

- GPIO_PU[7:0]
- GEN1_I2C_SDA
- GEN1_I2C_SCL

DBGCFG Pad Control Register

Offset: 894h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_DBGCFG_CAL_DRVUP_SLWF

Bit	Reset	Description
29:28	0x3	CFG2TMC_DBGCFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_DBGCFG_CAL_DRVUP
16:12	0x12	CFG2TMC_DBGCFG_CAL_DRVDN
5:4	0x3	CFG2TMC_DBGCFG_LPMD: DBGCFG data pins low power mode select
3	0x0	CFG2TMC_DBGCFG_SCHMT_EN: DBGCFG data pins Schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_DBGCFG_HSM_EN: DBGCFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

11.1.7.13 APB_MISC_GP_LCDCFG1PADCTRL_0

Controls the following pins:

- LCD_PWR1
- LCD_PWR2
- LCD_SDIN
- LCD_SDOUT
- LCD_WR_
- LCD_CS0_
- LCD_DC0
- LCD_SCK

LCDCFG1 Pad Control Register

Offset: 898h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_LCDCFG1_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_LCDCFG1_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_LCDCFG1_CAL_DRVUP
16:12	0x12	CFG2TMC_LCDCFG1_CAL_DRVDN
5:4	0x3	CFG2TMC_LCDCFG1_LPMD: LCDCFG1 data pins low power mode select
3	0x0	CFG2TMC_LCDCFG1_SCHMT_EN: LCDCFG1 data pins Schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_LCDCFG1_HSM_EN: LCDCFG1 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

11.1.7.14 APB_MISC_GP_LCDCFG2PADCTRL_0

Controls the following pins:

- LCD_PWR0
- LCD_PCLK
- LCD_DE
- LCD_HSYNC
- LCD_VSYNC
- LCD_D[23:0]
- LCD_CS1_
- LCD_M1
- LCD_DC1

LCDCFG2 Pad Control Register

Offset: 89ch | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_LCDCFG2_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_LCDCFG2_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_LCDCFG2_CAL_DRVUP
16:12	0x12	CFG2TMC_LCDCFG2_CAL_DRVDN
5:4	0x3	CFG2TMC_LCDCFG2_LPMD: LCDCFG2 data pins low power mode select
3	0x0	CFG2TMC_LCDCFG2_SCHMT_EN: LCDCFG2 data pins Schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_LCDCFG2_HSM_EN: LCDCFG2 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

11.1.7.15 APB_MISC_GP_SDIO2CFGPADCTRL_0

Controls following pins:

- SDIO2_DAT[3:0]
- SDIO2_CLK
- SDIO2_CMD

SDIO2CFG Pad Control Register

Offset: 8a0h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_SDIO2CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_SDIO2CFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_SDIO2CFG_CAL_DRVUP

Bit	Reset	Description
16:12	0x12	CFG2TMC_SDIO2CFG_CAL_DRVDN
5:4	0x3	CFG2TMC_SDIO2CFG_LPMD: SDIO2CFG data pins low power mode select
3	0x0	CFG2TMC_SDIO2CFG_SCHMT_EN: SDIO2CFG data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_SDIO2CFG_HSM_EN: SDIO2CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

11.1.7.16 APB_MISC_GP_SDIO3CFGPADCTRL_0

Controls the following pins:

- SDIO3_DAT[3:0]
- SDIO3_CLK
- SDIO3_CMD
- GPIO_PV[6:4]

SDIO3CFG Pad Control Register

Offset: 8a4h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_SDIO3CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_SDIO3CFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_SDIOCFG_CAL_DRVUP
16:12	0x12	CFG2TMC_SDIO3CFG_CAL_DRVDN
5:4	0x3	CFG2TMC_SDIO3CFG_LPMD: SDIO3CFG data pins low power mode select
3	0x0	CFG2TMC_SDIO3CFG_SCHMT_EN: SDIO3CFG data pins Schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_SDIO3CFG_HSM_EN: SDIO3CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

11.1.7.17 APB_MISC_GP_SPICFGPADCTRL_0

Controls the following pins:

- SPI2_MOSI
- SPI2_MISO
- SPI2_SCK
- SPI2_CS0_
- SPI1_MOSI
- SPI1_SCK

- SPI1_CS0_
- SPI1_MISO
- SPI2_CS1_
- SPI2_CS2_

SPICFG Pad Control Register

Offset: 8a8h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_SPICFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_SPICFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_SPICFG_CAL_DRVUP
16:12	0x12	CFG2TMC_SPICFG_CAL_DRVDN
5:4	0x3	CFG2TMC_SPICFG_LPMD: SPICFG data pins low power mode select
3	0x0	CFG2TMC_SPICFG_SCHMT_EN: SPICFG data pins Schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_SPICFG_HSM_EN: SPICFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

11.1.7.18 APB_MISC_GP_UAACFGPADCTRL_0

Controls the following pins:

- UART1_TXD
- UART1_RXD
- UART1_CTS_
- UART1_RTS_

UAACFG Pad Control Register

Offset: 8ach | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_UAACFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_UAACFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_UAACFG_CAL_DRVUP
16:12	0x12	CFG2TMC_UAACFG_CAL_DRVDN
5:4	0x3	CFG2TMC_UAACFG_LPMD: UAACFG data pins low power mode select
3	0x0	CFG2TMC_UAACFG_SCHMT_EN: UAACFG data pins Schmidt enable 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	0x0	CFG2TMC_UAACFG_HSM_EN: UAACFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

11.1.7.19 APB_MISC_GP_UABCFGPADCTRL_0

Controls the following pins:

- UART1_RI_
- UART1_DCD_
- UART1_DSR_
- UART1_DTR_
- GPIO_PV[3:0]

UABCFG Pad Control Register

Offset: 8b0h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_UABCFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_UABCFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_UABCFG_CAL_DRVUP
16:12	0x12	CFG2TMC_UABCFG_CAL_DRVDN
5:4	0x3	CFG2TMC_UABCFG_LPMD: UABCFG data pins low power mode select
3	0x0	CFG2TMC_UABCFG_SCHMT_EN: UABCFG data pins Schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_UABCFG_HSM_EN: UABCFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

11.1.7.20 APB_MISC_GP_UART2CFGPADCTRL_0

Controls the following pins:

- UART2_TXD
- UART2_RXD
- UART2_RTS_
- UART2_CTS_

UART2CFG Pad Control Register

Offset: 8b4h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_UART2CFG_CAL_DRVUP_SLWF

Bit	Reset	Description
29:28	0x3	CFG2TMC_UART2CFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_UART2CFG_CAL_DRVUP
16:12	0x12	CFG2TMC_UART2CFG_CAL_DRVDN
5:4	0x3	CFG2TMC_UART2CFG_LPMD: UART2CFG data pins low power mode select
3	0x0	CFG2TMC_UART2CFG_SCHMT_EN: UART2CFG data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_UART2CFG_HSM_EN: UART2CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

11.1.7.21 APB_MISC_GP_UART3CFGPADCTRL_0

Controls the following pins:

- UART3_TXD
- UART3_RXD
- UART3_RTS_
- UART3_CTS_

UART3CFG Pad Control Register

Offset: 8b8h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_UART3CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_UART3CFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_UART3CFG_CAL_DRVUP
16:12	0x12	CFG2TMC_UART3CFG_CAL_DRVDN
5:4	0x3	CFG2TMC_UART3CFG_LPMD: UART3CFG data pins low power mode select
3	0x0	CFG2TMC_UART3CFG_SCHMT_EN: UART3CFG data pins Schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_UART3CFG_HSM_EN: UART3CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

11.1.7.22 APB_MISC_GP_VICFG1PADCTRL_0

Controls the following pins:

- VI_D[11:0]
- VI_PCLK
- VI_VSYNC
- VI_HSYNC

VICFG1 Pad Control Register

Offset: 8bch | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_VICFG1_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_VICFG1_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_VICFG1_CAL_DRVUP
16:12	0x12	CFG2TMC_VICFG1_CAL_DRVDN
5:4	0x3	CFG2TMC_VICFG1_LPMD: VICFG1 data pins low power mode select
3	0x0	CFG2TMC_VICFG1_SCHMT_EN: VICFG1 data pins Schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_VICFG1_HSM_EN: VICFG1 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

11.1.7.23 APB_MISC_GP_VICFG2PADCTRL_0

Controls the following pins:

- VI_GP0
- CAM_I2C_SCK
- CAM_I2C_SDA
- VI_GP[6:3]

VICFG2 Pad Control Register

Offset: 8c0h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_VICFG2_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_VICFG2_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_VICFG2_CAL_DRVUP
16:12	0x12	CFG2TMC_VICFG2_CAL_DRVDN
5:4	0x3	CFG2TMC_VICFG2_LPMD: VICFG2 data pins low power mode select
3	0x0	CFG2TMC_VICFG2_SCHMT_EN: VICFG2 data pins Schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_VICFG2_HSM_EN: VICFG2 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

11.1.7.24 APB_MISC_GP_XM2CFGAPADCTRL_0

Controls the following pins:

- SPI3_SCK
- SPI3_DOUT
- SPI3_DIN
- SPI3_CS0_
- DDR_A [13: 0]
- DDR_BA0
- DDR_BA1

XM2CFGA Pad Control Register

Offset: 8c4h | Read/Write: R/W | Reset: 0b11111111111111111xxxxxxx000

Bit	Reset	Description
31:28	0xf	CFG2TMC_XM2CFGA_CAL_DRVUP_SLWF
27:24	0xf	CFG2TMC_XM2CFGA_CAL_DRVDN_SLWR
23:19	0x1f	CFG2TMC_XM2CFGA_CAL_DRVUP
18:14	0x1f	CFG2TMC_XM2CFGA_CAL_DRVDN
6	0x0	CFG2TMC_XM2CFGA_CLK_SEL: pad clk_sel (ma bits get this value inverted in lpddr2 mode)
5	0x0	CFG2TMC_XM2CFGA_PREEMP_EN: XM2CFGA data pins preemp enable 0 = DISABLE 1 = ENABLE
4	0x0	CFG2TMC_XM2CFGA_BYPASS_EN: XM2CFGA data pins bypass outbound flop enable 0 = DISABLE 1 = ENABLE

11.1.7.25 APB_MISC_GP_XM2CFGCPADCTRL_0

Controls the following pins:

- MIO_IORDY
- ROM_CS0_
- MIO_CS0_
- DDR_DQS0
- DDR_DQS1
- DDR_DQS2
- DDR_DQS3
- DDR_DM0
- DDR_DM1
- DDR_DM2
- DDR_DM3
- DDR_CS0_

- DDR_CS1_
- DDR_CKE
- DDR_RAS_
- DDR_CAS_
- DDR_WE_

XM2CFGC Pad Control Register

Offset: 8c8h | Read/Write: R/W | Reset: 0b11111111111111111111111111110

Bit	Reset	Description
31:28	0xf	CFG2TMC_XM2CFGC_CAL_DRVUP_SLWF
27:24	0xf	CFG2TMC_XM2CFGC_CAL_DRVDN_SLWR
23:19	0x1f	CFG2TMC_XM2CFGC_CAL_DRVUP
18:14	0x1f	CFG2TMC_XM2CFGC_CAL_DRVDN
13:9	0x1f	CFG2TMC_XM2CFGC_CAL_DRVUP_TERM
8:4	0x1f	CFG2TMC_XM2CFGC_CAL_DRVDN_TERM
3	0x0	CFG2TMC_XM2CFGC_SCHMT_EN: XM2CFGD data pins schmidt enable 0 = DISABLE 1 = ENABLE

11.1.7.26 APB_MISC_GP_XM2CFGDPADCTRL_0

Controls the following pins:

- DDR_DQ[31:0]

XM2CFGD Pad Control Register

Offset: 8cch | Read/Write: R/W | Reset: 0b11111111111111111111111111110

Bit	Reset	Description
31:28	0xf	CFG2TMC_XM2CFGD_CAL_DRVUP_SLWF
27:24	0xf	CFG2TMC_XM2CFGD_CAL_DRVDN_SLWR
23:19	0x1f	CFG2TMC_XM2CFGD_CAL_DRVUP
18:14	0x1f	CFG2TMC_XM2CFGD_CAL_DRVDN
13:9	0x1f	CFG2TMC_XM2CFGD_CAL_DRVUP_TERM
8:4	0x1f	CFG2TMC_XM2CFGD_CAL_DRVDN_TERM
3	0x0	CFG2TMC_XM2CFGD_SCHMT_EN: XM2CFGD data pins schmidt enable 0 = DISABLE 1 = ENABLE

11.1.7.27 APB_MISC_GP_XM2CLKCFGPADCTRL_0

Controls the following pins:

- DDR_CLK
- DDR_CLK_
- DDR_CKE

XM2CLKCFG Pad Control Register

Offset: 8d0h | Read/Write: R/W | Reset: 0b11111111111111111xxxxxxxxxx001

Bit	Reset	Description
31:28	0xf	CFG2TMC_XM2CLKCFG_CAL_DRVUP_SLWF
27:24	0xf	CFG2TMC_XM2CLKCFG_CAL_DRVDN_SLWR
23:19	0x1f	CFG2TMC_XM2CLKCFG_CAL_DRVUP
18:14	0x1f	CFG2TMC_XM2CLKCFG_CAL_DRVDN
3	0x0	CFG2TMC_XM2CLKCFG_CAL_BYPASS_EN: XM2CLKCFG bypass drvdn/up calibration 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_XM2CLKCFG_PREEMP_EN: preemp enable 0 = DISABLE 1 = ENABLE
1	0x1	CFG2TMC_XM2CLKCFG_BYPASS_EN: XM2 bypass outbound flop enable 0 = DISABLE 1 = ENABLE

11.1.7.28 APB_MISC_GP_XM2COMPPADCTRL_0

MEM_COMP Pad Control Registers

Offset: 8d4h | Read/Write: R/W | Reset: 0b11111xxx11111xxxx00001000

Bit	Reset	Description
24:20	0x1f	CFG2TMC_XM2COMP_DRVUP
16:12	0x1f	CFG2TMC_XM2COMP_DRVDN
7:5	0x0	CFG2TMC_XM2COMP_BIAS_SEL
4	0x0	CFG2TMC_XM2COMP_TESTOUT_EN: 0 = DISABLE 1 = ENABLE
3:0	0x8	CFG2TMC_XM2COMP_VREF_SEL

11.1.7.29 APB_MISC_GP_XM2VTTGENPADCTRL_0

XM2 MISC/VTTGEN Pad control register

Offset: 8d8h | Read/Write: R/W | Reset: 0b 000xxxxx000x101x101xxxxxxx0

Bit	Reset	Description
26:24	0x0	CFG2TMC_XM2VTTGEN_CAL_DRVUP
18:16	0x0	CFG2TMC_XM2VTTGEN_CAL_DRVDN
14:12	0x5	CFG2TMC_XM2VTTGEN_VAUXP_LEVEL
10:8	0x5	CFG2TMC_XM2VTTGEN_VCLAMP_LEVEL
1	X	CFG2TMC_XM2VTTGEN_SHORT_PWRGND: dummy pin
0	0x0	CFG2TMC_XM2VTTGEN_SHORT

11.1.7.30 APB_MISC_GP_PADCTL_DFT_0

DFT Pin shorting

Offset: 8dch | Read/Write: R/W | Reset: 0b 00

Bit	Reset	Description
1	0x0	PINSHORT_SEL: Select which pins are used for test-mode observe
0	0x0	PINSHORT_EN: Enable pin-shorting for tester mode pin-shorting 0 = DISABLE 1 = ENABLE

11.1.7.31 APB_MISC_GP_SDIO1CFGPADCTRL_0

SDIO1CFG Pad control register

Offset: 8e0h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_SDIO1CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_SDIO1CFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_SDIO1CFG_CAL_DRVUP
16:12	0x12	CFG2TMC_SDIO1CFG_CAL_DRVDN
5:4	0x3	CFG2TMC_SDIO1CFG_LPMD: SDIO3CFG data pins low power mode select
3	0x0	CFG2TMC_SDIO1CFG_SCHMT_EN: SDIO3CFG data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_SDIO1CFG_HSM_EN: SDIO3CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

11.1.7.32 APB_MISC_GP_XM2CFGCPADCTRL2_0

XM2CFGC Pad control register

Offset: 8e4h | Read/Write: R/W | Reset: 0b1000xxxx1000xxxxxxxx01000010

Bit	Reset	Description
27:24	0x8	CFG2TMC_XM2CFGC_VREF_DQ
19:16	0x8	CFG2TMC_XM2CFGC_VREF_DQS
7	0x0	CFG2TMC_XM2CFGC_CLKSEL_DQS
6	0x1	CFG2TMC_XM2CFGC_CLKSEL_DQ
5	0x0	CFG2TMC_XM2CFGC_VREF_DQ_EN: 0 = DISABLE 1 = ENABLE
4	0x0	CFG2TMC_XM2CFGC_VREF_DQS_EN: 0 = DISABLE 1 = ENABLE
3	0x0	CFG2TMC_XM2CFGC_CTT_HIZ_EN: 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_XM2CFGC_PREEMP_EN: XM2CFGD data pins preemp enable 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1	0x1	CFG2TMC_XM2CFGC_BYPASS_EN: XM2CFGD data pins bypass outbound flop enable 0 = DISABLE 1 = ENABLE
0	0x0	CFG2TMC_XM2CFGC_RX_FT_REC_EN: 0 = DISABLE 1 = ENABLE

11.1.7.33 APB_MISC_GP_XM2CFGDPADCTRL2_0

XM2CFGD Pad control register

Offset: 8e8h | Read/Write: R/W | Reset: 0b000x000x000x000xxxxxxxxxxxx0010

Bit	Reset	Description
30:28	0x0	CFG2TMC_XM2CFGD3_DLYIN_TRM: delay trim for byte 3
26:24	0x0	CFG2TMC_XM2CFGD2_DLYIN_TRM: delay trim for byte 2
22:20	0x0	CFG2TMC_XM2CFGD1_DLYIN_TRM: delay trim for byte 1
18:16	0x0	CFG2TMC_XM2CFGD0_DLYIN_TRM: delay trim for byte 0
3	0x0	CFG2TMC_XM2CFGD_CTT_HIZ_EN: 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_XM2CFGD_PREEMP_EN: XM2CFGD data pins preemp enable 0 = DISABLE 1 = ENABLE
1	0x1	CFG2TMC_XM2CFGD_BYPASS_EN: XM2CFGD data pins bypass outbound flop enable 0 = DISABLE 1 = ENABLE
0	0x0	CFG2TMC_XM2CFGD_RX_FT_REC_EN: 0 = DISABLE 1 = ENABLE

11.1.7.34 APB_MISC_GP_CRTCFGPADCTRL_0

CRTCFG Pad control register

Offset: 8ech | Read/Write: R/W | Reset: 0b000x000x000x000xxxxxxxxxxxx0010

Bit	Reset	Description
31:30	0x3	CFG2TMC_CRTCFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_CRTCFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_CRTCFG_CAL_DRVUP
16:12	0x12	CFG2TMC_CRTCFG_CAL_DRVDN
5:4	0x3	CFG2TMC_CRTCFG_LPMD: SDIO3CFG data pins low power mode select
3	0x0	CFG2TMC_CRTCFG_SCHMT_EN: SDIO3CFG data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_CRTCFG_HSM_EN: SDIO3CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

11.1.7.35 APB_MISC_GP_DDCCFGPADCTRL_0

DDCCFG Pad control register

Offset: 8f0h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_DDCCFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_DDCCFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_DDCCFG_CAL_DRVUP
16:12	0x12	CFG2TMC_DDCCFG_CAL_DRVDN
5:4	0x3	CFG2TMC_DDCCFG_LPMD: SDIO3CFG data pins low power mode select
3	0x0	CFG2TMC_DDCCFG_SCHMT_EN: SDIO3CFG data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_DDCCFG_HSM_EN: SDIO3CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

11.1.7.36 APB_MISC_GP_GMACFGPADCTRL_0

GMACFG Pad control register

Offset: 8f4h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_GMACFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_GMACFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_GMACFG_CAL_DRVUP
16:12	0x12	CFG2TMC_GMACFG_CAL_DRVDN
5:4	0x3	CFG2TMC_GMACFG_LPMD: SDIO3CFG data pins low power mode select
3	0x0	CFG2TMC_GMACFG_SCHMT_EN: SDIO3CFG data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_GMACFG_HSM_EN: SDIO3CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

11.1.7.37 APB_MISC_GP_GMBCFGPADCTRL_0

GMBCFG Pad control register

Offset: 8f8h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx110

Bit	Reset	Description
31:30	0x3	CFG2TMC_GMBCFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_GMBCFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_GMBCFG_CAL_DRVUP
16:12	0x12	CFG2TMC_GMBCFG_CAL_DRVDN
5:4	0x3	CFG2TMC_GMBCFG_LPMD: SDIO3CFG data pins low power mode select

Bit	Reset	Description
3	0x0	CFG2TMC_GMBCFG_SCHMT_EN: SDIO3CFG data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_GMBCFG_HSM_EN: SDIO3CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

11.1.7.38 APB_MISC_GP_GMCCFGPADCTRL_0

GMCCFG Pad control register

Offset: 8fch | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_GMCCFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_GMCCFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_GMCCFG_CAL_DRVUP
16:12	0x12	CFG2TMC_GMCCFG_CAL_DRVDN
5:4	0x3	CFG2TMC_GMCCFG_LPMD: SDIO3CFG data pins low power mode select
3	0x0	CFG2TMC_GMCCFG_SCHMT_EN: SDIO3CFG data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_GMCCFG_HSM_EN: SDIO3CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

11.1.7.39 APB_MISC_GP_GMDCFGPADCTRL_0

GMDCFG Pad control register

Offset: 900h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_GMDCFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_GMDCFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_GMDCFG_CAL_DRVUP
16:12	0x12	CFG2TMC_GMDCFG_CAL_DRVDN
5:4	0x3	CFG2TMC_GMDCFG_LPMD: SDIO3CFG data pins low power mode select
3	0x0	CFG2TMC_GMDCFG_SCHMT_EN: SDIO3CFG data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_GMDCFG_HSM_EN: SDIO3CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

11.1.7.40 APB_MISC_GP_GMECFGPADCTRL_0

GMECFG Pad control register

Offset: 904h | Read/Write: R/W | Reset: 0 b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_GMECFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_GMECFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_GMECFG_CAL_DRVUP
16:12	0x12	CFG2TMC_GMECFG_CAL_DRVDN
5:4	0x3	CFG2TMC_GMECFG_LPMD: SDIO3CFG data pins low power mode select
3	0x0	CFG2TMC_GMECFG_SCHMT_EN: SDIO3CFG data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_GMECFG_HSM_EN: SDIO3CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

11.1.7.41 APB_MISC_GP_OWRCFGPADCTRL_0

OWRCFG Pad control register

Offset: 908h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_OWRCFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_OWRCFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_OWRCFG_CAL_DRVUP
16:12	0x12	CFG2TMC_OWRCFG_CAL_DRVDN
5:4	0x3	CFG2TMC_OWRCFG_LPMD: SDIO3CFG data pins low power mode select
3	0x0	CFG2TMC_OWRCFG_SCHMT_EN: SDIO3CFG data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_OWRCFG_HSM_EN: SDIO3CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

11.1.7.42 APB_MISC_GP_UADCFGPADCTRL_0

UDACFG Pad control register

Offset: 90ch | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_UDACFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_UDACFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_UDACFG_CAL_DRVUP
16:12	0x12	CFG2TMC_UDACFG_CAL_DRVDN
5:4	0x3	CFG2TMC_UDACFG_LPMD: SDIO3CFG data pins low power mode select

Bit	Reset	Description
3	0x0	CFG2TMC_UDACFG_SCHMT_EN: SDIO3CFG data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_UDACFG_HSM_EN: SDIO3CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

11.2 APB DMA Controller

The APB DMA Controller is placed between the AHB Bus and the APB Bus and is a master on both buses.

The APB DMA is used for block data transfer from a source location to the destination location. The source may be DRAM or IRAM, and the destination location could be devices placed on APB Bus; or *vice versa*. DMA transfers are done without any processor intervention other than register writes to program the parameters for a particular transfer and accesses needed to handle any interrupts.

It has sixteen fully programmable channels, which can transfer data concurrently.

Hardware Features

- DMA master for transfers between external/internal memory and peripheral devices on the APB Bus
- Two modes of operation: single transfer (once) or continuous
- Programmable burst sizes of 1, 4 or 8 words
- Max transfer size is 64KB per channel, with the minimum size being one word
- Programmable APB Bus widths of 8, 16 and 32 bits
- Separate AHB and APB start addresses
- Per channel trigger and flow control mechanism support
- Channel to channel trigger support, i.e., ability to link up channels to start at the end of another channel's transfer, allowing scattering/gathering of physical memory.
- Interrupt generation at the completion of channel transfer
- Interrupts per channel can be routed to CPU or AVP
- Ability to hold processor until transfer is done
- Wrap mode supported for all channels in Once mode
- Ping-Pong feature supported in continuous mode
- Round robin arbitration among channels at burst granularity
- Runs on APB clock. APB Clock generally runs at 1:2:2 or 1:2:3 clock ratios (sclk: hclk : pclk).

Software Features

- The APB burst size and the FIFO trigger levels (in the modules) should be programmed such that they do not lead to an overflow/underflow of the FIFO.
- SW should program all the registers of channel, ensuring that the channel enable bit is disabled. Channel enable bit should be set last.
- Disable the Global enable bit to halt transfer for some time, and set it when it is all right to resume the transfer. This halts transfers from all channels.
- If channel is disabled while transfer is in progress, the transfer ends after ongoing burst is completed and an interrupt is generated if enabled.
- Busy bit is set as soon as DMA channel is enabled and gets cleared after transfer is completed.
- Interrupts are "write 1 to clear".
- Interrupt is generated when the last data is placed on the AHB bus while writing and once the last data is placed on APB bus while reading from AHB bus.
- The AHB wrap around should be programmed keeping in mind the address that is programmed in the AHB start address and the burst size.

- If, for example, AHB wrap around is 4 words, the AHB start address is 0x4000 0000 and the AHB burst size is NOT 1, then address would change to:

  0x4000 0000 ' 0x4000 0004 ' 0x4000 0008 ' 0x4000 000C ' 0x4000 0000

- For the above condition if the AHB start address is programmed to

  0x0000 0004….

- Then the address would change as

  0x0000 0004 ' 0x0000 0008 ' 0x0000 000C ' 0x0000 0010

  Note that the address did not wrap around! Wrap around is NOT possible in between bursts.

- If this burst size is given to the AHB bus it increments the address 4 times as it does not see "AHB current address" but the initial address that was given to it with a request.
- If the start address has to be specified as 0x0000 0004 with wrap around as needed above, the solution is to give a burst size of 1.

- The default wrap around on the APB side is wrapping on 1 word. It prevents unnecessary address switching on the APB.
- The parameters of the channel should be programmed first (for e.g., base addr, wrap-around). The control register of that channel should then be programmed. If the control register is programmed first, the current parameters of the DMA would be considered as the programmed values.
- If a burst of 8 is requested with the address that is not aligned to a 8-word boundary (addr [4:2] not zeros), the DMA does a single burst till it aligns itself to the 8-word boundary and then starts issuing burst requests (on AHB address).
- If a burst of 8 is requested and at any point of time the transfer size goes below 8 (less than 8 words left to transfer), the DMA completes the remaining transfers with a burst of 4 or 1, whichever is possible.
- If a burst of 4 is requested with the address that is not aligned to a 4-word boundary (addr [3:2] not zeros), the DMA does a single burst till it aligns itself to the 4-word boundary.
- If a burst of 4 is requested and at any point of time the transfer size goes below 4 (less than 4 words left to transfer), the DMA completes the remaining transfers with a burst of 1.

11.2.1 Functionality

There are 16 channels in APB DMA. An APB DMA channel can transfer specified portions of data from an AHB address space (can be iRAM's or DRAM's on MC) to an APB address space. APB DMA follows a simple round robin arbitration scheme, starting with ch-0.

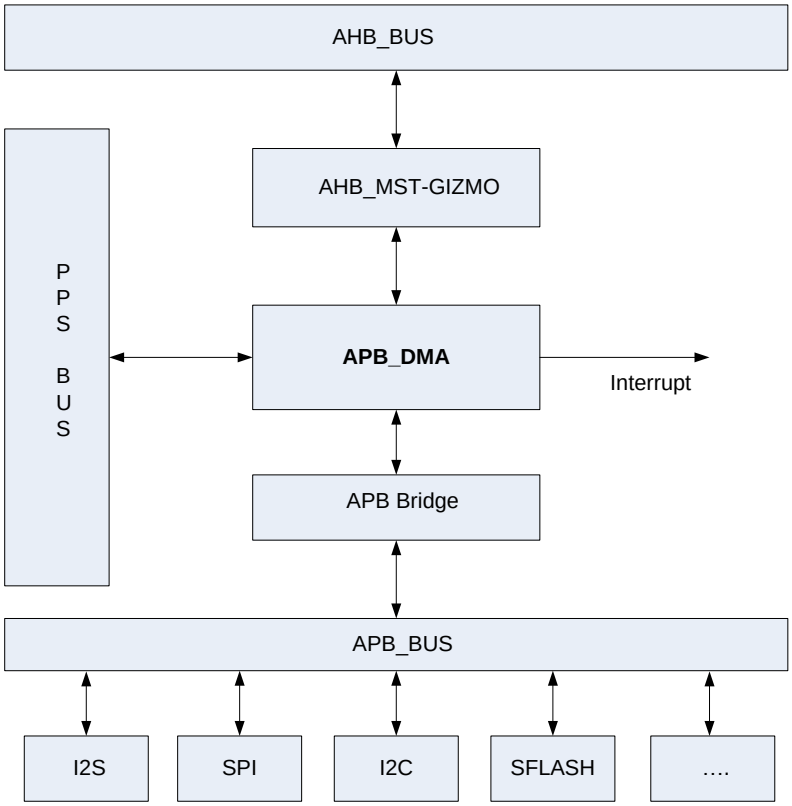
Each channel can have independent burst transfer sizes programmed to one word, four words, or eight words.

Each DMA channel supports:

- Enable Bit one for each channel and one global enable bit.
- Interrupt Enable at the end of transfer with ability to mask or route to desired processor.
- Ability to hold off a Processor. Processor which writes into this bit will be held until transfer is completed.
- Direction bit to determine the direction of transfer--AHB to APB or APB to AHB.
- Trigger and Flow selects. These are addition controls apart from channel enable, on which the transfer depends. Trigger is used to start a channel on some event to start the transfer and flow is used to proceed with every new burst transfer. These events are under either Software or Hardware control.

- Wrap feature enables you to wrap back the LSB nibble to 0000 instead of incrementing to next address if the burst starts from an unaligned address boundary. For example, if address starts at 0x4 with a burst of 4 words, then the address would increment as 0x4, 0x8, 0xC, 0x0. If disabled, it would be 0x4, 0x8, 0xC, 0x10.
- APB bus width is configurable whereas the AHB bus width is fixed to 32 bits. The APB bus can have values 8, 16 and 32. For an APB burst of 8 (burst is always with regard to words) and APB bus size of 16, you have: 8 words transferred to the APB side, or rather 16 "half words" transferred.
- Double Buffering Mode makes the APB DMA Burst Address reset to AHB Base Address after every even (2nd, 4th, 6th, etc) APB DMA Transfer. This mode is used only along with continuous mode (once bit 0).
- Separate AHB start address and APB start address.
- Ability to delay the burst rate with the help of a global counter which can be programmed to desired value, which equals number of clocks delay required between each burst.

Figure 16. APB-DMA Block Diagram



11.2.2 Programming Guidelines

Follow these guidelines when programming the APB_DMA Controller:

1. All the registers of a channel should be programmed before the Channel Enable bit is set. To program each register:
   a. Program the AHB Starting Address and APB Starting address in *AHB_PTR and *APB_PTR Registers.
   b. Program the required AHB BURST size, WRAP word window size and AHB_DATA_SWAP (byte swapping) option in *AHB_SWQ Register. The AHB BUS WIDTH is fixed to 32 bit Bus.
   c. Program the required APB_BUS_WIDTH (as the peripheral), APB_DATA_SWAP (byte swapping) option and WRAP word window size in *APB_SEQ Register.

d. Program the Interrupt option, Hold Processor option, DMA transfer direction, Transfer Mode, Trigger/Flow Enable and DMA Count value in CHANNEL_*_CSR Register.
e. Program the Global Enable bit in APB_DMA Command Register.
f. Whenever the Channel ENB bit is Enabled, the DMA starts the Data transfer.
g. Each channel status is observed by polling or interrupt status using APB_DMA Status Register.
h. The Tx/Rx Flow/Trigger requestors are programmed in APB-DMA Requestor Assignments Registers.

2. Clearing the Global Enable bit causes the transfers that are in progress to be paused. Setting the Global Enable resumes the transfers.
3. If channel ENB is disabled independently while a transfer is in progress, the transfer ends after completing any burst sequence that is in progress.
4. Busy bit gets set as soon as a channel is enabled and is cleared after transfer is completed.
5. Interrupts are write-1-to-clear i.e., interrupt bit is cleared when the value of write data corresponding to the bit position of the interrupt bit is 1.
6. The 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.
7. The AHB wrap around needs to be programmed keeping in mind the address that is programmed in the AHB start address and the burst size.
8. If a burst of 8 is requested with the address that is not aligned to a 8-word boundary, the DMA would do a single burst till it aligns itself to the 8-word boundary and then starts issuing burst requests.
9. If a burst of 8 is requested and at any point of time the transfer size goes below 8, the DMA completes the remaining transfers with a burst of 4 or 1, whichever is possible.
10. If a burst of 4 is requested with the address that is not aligned to a 4-word boundary, the DMA does a single burst till it aligns itself to the 4-word boundary.
11. If a burst of 4 is requested and at any time the transfer size goes below 4, the DMA completes the remaining transfers with a burst of 1.

11.2.3 APB DMA Registers

11.2.3.1 APBDMA_COMMAND_0

The Global Enable bit in the command register enables the APB DMA. Clearing this bit causes active DMA transfers to be paused. Pending bus transactions (ongoing burst) will be completed and no new transactions will be initiated. Setting this bit again resumes the transfers.

Note that the power on reset value for the APB DMA Global Enable is 0. This bit must be written to 1 before any APB DMA transactions can begin.

APB-DMA Command Register

Offset: 000h | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
31	0x0	GEN: Enables Global APB-DMA 0 = DISABLE 1 = ENABLE

11.2.3.2 APBDMA_STATUS_0

The Busy bits in the Status Register indicate which (if any) of the APB DMA channels have active pending APB DMA transfers. Note that APB DMA transfers that are started in continuous (repetitive) mode will have their busy bits active until the enable bit for the APB DMA channel is cleared to 0 (by the software).

Read-only Flags set/cleared by HW.

APB-DMA Status Register

Offset: 004h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	BSY_15: DMA channel15 status 0 = NOT_BUSY 1 = BUSY
30	X	BSY_14: DMA channel14 status 0 = NOT_BUSY 1 = BUSY
29	X	BSY_13: DMA channel13 status 0 = NOT_BUSY 1 = BUSY
28	X	BSY_12: DMA channel12 status 0 = NOT_BUSY 1 = BUSY
27	X	BSY_11: DMA channel11 status 0 = NOT_BUSY 1 = BUSY
26	X	BSY_10: DMA channel10 status 0 = NOT_BUSY 1 = BUSY
25	X	BSY_9: DMA channel9 status 0 = NOT_BUSY 1 = BUSY
24	X	BSY_8: DMA channel8 status 0 = NOT_BUSY 1 = BUSY
23	X	BSY_7: DMA channel7 status 0 = NOT_BUSY 1 = BUSY
22	X	BSY_6: DMA channel6 status 0 = NOT_BUSY 1 = BUSY
21	X	BSY_5: DMA channel5 status 0 = NOT_BUSY 1 = BUSY
20	X	BSY_4: DMA channel4 status 0 = NOT_BUSY 1 = BUSY

Bit	Reset	Description
19	X	BSY_3: DMA channel3 status 0 = NOT_BUSY 1 = BUSY
18	X	BSY_2: DMA channel2 status 0 = NOT_BUSY 1 = BUSY
17	X	BSY_1: DMA channel1 status 0 = NOT_BUSY 1 = BUSY
16	X	BSY_0: DMA channel0 status 0 = NOT_BUSY 1 = BUSY
15	X	ISE_EOC_15: DMA channel15 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
14	X	ISE_EOC_14: DMA channel14 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
13	X	ISE_EOC_13: DMA channel13 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
12	X	ISE_EOC_12: DMA channel12 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
11	X	ISE_EOC_11: DMA channel11 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
10	X	ISE_EOC_10: DMA channel10 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
9	X	ISE_EOC_9: DMA channel9 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
8	X	ISE_EOC_8: DMA channel8 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
7	X	ISE_EOC_7: DMA channel7 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
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

Bit	Reset	Description
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

11.2.3.3 APBDMA_REQUESTORS_TX_0

The Tx Requestors are values used as triggers or flow controls for APB DMA transfers.

The Tx requestor register is read only. This register is provided only to give the software the ability to see the state of signals that may affect (trigger-start or flow-control-inhibit) APB DMA processes.

Bit[0] in the Tx Requestor channel is a signal generated from the programmable APB DMA Counter. This bit is active whenever the APB DMA Counter value is 0.

APB-DMA Requestor Register (Tx)

Offset: 008h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25	X	OWR: OWR-I2C 0 = NOT_ACTIVE 1 = ACTIVE
24	X	DVC_I2C: DVC-I2C 0 = NOT_ACTIVE 1 = ACTIVE
23	X	I2C_3: I2C3 0 = NOT_ACTIVE 1 = ACTIVE
22	X	I2C_2: I2C2 0 = NOT_ACTIVE 1 = ACTIVE
21	X	I2C_1: I2C1 0 = NOT_ACTIVE 1 = ACTIVE
20	X	UART_E: UART5 0 = NOT_ACTIVE 1 = ACTIVE

Bit	Reset	Description
19	X	UART_C: UART4 0 = NOT_ACTIVE 1 = ACTIVE
18	X	SL2B4: SLINK 2B-4 (SPI4) 0 = NOT_ACTIVE 1 = ACTIVE
17	X	SL2B3: SLINK 2B-3 (SPI3) 0 = NOT_ACTIVE 1 = ACTIVE
16	X	SL2B2: SLINK 2B-2 (SPI2) 0 = NOT_ACTIVE 1 = ACTIVE
15	X	SL2B1: SLINK 2B-1 (SPI1) 0 = NOT_ACTIVE 1 = ACTIVE
14	X	RSVD: 0 = NOT_ACTIVE 1 = ACTIVE
13	X	AC Modem: AC Modem 0 = NOT_ACTIVE 1 = ACTIVE
12	X	AC97: AC97 0 = NOT_ACTIVE 1 = ACTIVE
11	X	SPI: SPI Controller 0 = NOT_ACTIVE 1 = ACTIVE
10	X	UART_C: UART 3 0 = NOT_ACTIVE 1 = ACTIVE
9	X	UART_B: UART 2 (VFIR) 0 = NOT_ACTIVE 1 = ACTIVE
8	X	UART_A: UART 1 0 = NOT_ACTIVE 1 = ACTIVE
7	X	I2S2_1: I2S2 Tx Output FIFO1 (Play) (Peripheral initiated DMA request) 1 = Activate DMA transfer 0 = NOP 0 = NOT_ACTIVE 1 = ACTIVE
6	X	I2S2_2: I2S2 Tx Output FIFO2 (Play) (Peripheral initiated DMA request) 1 = Activate DMA transfer 0 = NOP 0 = NOT_ACTIVE 1 = ACTIVE
5	X	MIPI: MIPI Rx Input FIFO.

Bit	Reset	Description
4	X	UI_I: EBU USR Output (Peripheral initiated DMA request) 1 = Activate DMA transfer 0 = NOP 0 = NOT_ACTIVE 1 = ACTIVE
3	X	SPD_I: SPDIF Output FIFO (Rx) (Peripheral initiated DMA request) 1 = Activate DMA transfer 0 = NOP 0 = NOT_ACTIVE 1 = ACTIVE
2	X	I2S_1: I2S Tx Output FIFO1 (Record) (Peripheral initiated DMA request) 1 = Activate DMA transfer0 = NOP 0 = NOT_ACTIVE 1 = ACTIVE
1	X	I2S_2: I2S Tx Output FIFO2 (Play) (Peripheral initiated DMA request) 1 = Activate DMA transfer0 = NOP 0 = NOT_ACTIVE 1 = ACTIVE
0	X	CNTR_REQ: 1 = Enable counter request. 0 = Disable counter request 0 = NOT_ACTIVE 1 = ACTIVE

11.2.3.4 APBDMA_REQUESTORS_RX_0

The Rx Requestors are values used as triggers or flow controls for APB DMA transfers.

The Rx requestor register is read only. This register is provided only to give the software the ability to see the state of signals that may affect (trigger-start or flow-control-inhibit) APB DMA processes.

Bit[0] in the Rx Requestor channel is a signal generated from the programmable APB DMA Counter. This bit is active whenever the APB DMA Counter value is 0.

APB-DMA Requestor Register (RX)

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

Bit	Reset	Description
25	X	OWR: OWR-I2C 0 = NOT_ACTIVE 1 = ACTIVE
24	X	DVC_I2C: DVC-I2C 0 = NOT_ACTIVE 1 = ACTIVE
23	X	I2C_3: I2C3 0 = NOT_ACTIVE 1 = ACTIVE
22	X	I2C_2: I2C2 0 = NOT_ACTIVE 1 = ACTIVE
21	X	I2C_1: I2C1 0 = NOT_ACTIVE 1 = ACTIVE

Bit	Reset	Description
20	X	UART_E: UART5 0 = NOT_ACTIVE 1 = ACTIVE
19	X	UART_C: UART4 0 = NOT_ACTIVE 1 = ACTIVE
18	X	SL2B4: SLINK 2B-4 (SPI4) 0 = NOT_ACTIVE 1 = ACTIVE
17	X	SL2B3: SLINK 2B-3 (SPI3) 0 = NOT_ACTIVE 1 = ACTIVE
16	X	SL2B2: SLINK 2B-2 (SPI2) 0 = NOT_ACTIVE 1 = ACTIVE
15	X	SL2B1: SLINK 2B-1 (SPI1) 0 = NOT_ACTIVE 1 = ACTIVE
14	X	RSVD
13	X	AC Modem: AC Modem 0 = NOT_ACTIVE 1 = ACTIVE
12	X	AC97: AC97 0 = NOT_ACTIVE 1 = ACTIVE
11	X	SPI: SPI Controller 0 = NOT_ACTIVE 1 = ACTIVE
10	X	UART_C: UART 3 0 = NOT_ACTIVE 1 = ACTIVE
9	X	UART_B: UART 2 (VFIR) 0 = NOT_ACTIVE 1 = ACTIVE
8	X	UART_A: UART 1 0 = NOT_ACTIVE 1 = ACTIVE
7	X	I2S2_2: I2S2 Rx Input FIFO2 (Peripheral initiated DMA request) 1 = Activate DMA transfer 0 = NOP 0 = NOT_ACTIVE 1 = ACTIVE
6	X	I2S2_1: I2S2 Rx Input FIFO1 (Peripheral initiated DMA request) 1 = Activate DMA transfer 0 = NOP 0 = NOT_ACTIVE 1 = ACTIVE
5	X	MIPI: MIPI Rx Input FIFO.

Bit	Reset	Description
4	X	UI_I: EBU+SPDIF USR Input (Peripheral initiated DMA request) 1 = Activate DMA transfer 0 = NOP 0 = NOT_ACTIVE 1 = ACTIVE
3	X	SPD_I: SPDIF Input FIFO (Rx) (Peripheral initiated DMA request) 1 = Activate DMA transfer 0 = NOP 0 = NOT_ACTIVE 1 = ACTIVE
2	X	I2S_2: I2S Rx Input FIFO2 (Peripheral initiated DMA request) 1 = Activate DMA transfer 0 = NOP 0 = NOT_ACTIVE 1 = ACTIVE
1	X	I2S_1: I2S Rx Input FIFO1 (Peripheral initiated DMA request) 1 = Activate DMA transfer 0 = NOP 0 = NOT_ACTIVE 1 = ACTIVE
0	X	CNTR_REQ: indicates Enabled counter request or not 0 = NOT_ACTIVE 1 = ACTIVE

11.2.3.5 APBDMA_CNTRL_REG_0

The APB DMA Counter is used to slow down the request rates on some APB DMA channels.

The APB DMA Counter register stores bits that configure which (if any) channel(s) should be throttled (bits [31:16]) correspond to APB DMA 16 channels.

The APB DMA Counter initial/reload count value is programmed in the APB DMA Counter Register (bits[15:0]).The APB DMA Counter is loaded with this initial/reload count value whenever the APB DMA Counter Register is written, and is re-loaded to this saved initial count value on a burst complete (programmable).

The APB DMA Counter value will decrement whenever an APB DMA burst complete, and the current count value is non-zero, and any of the bits [31:16] are set.

APB-DMA Counter Register

Offset: 010h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	CH15_CNT_EN: Enable the channel15 count 0 = DISABLE 1 = ENABLE
30	0x0	CH14_CNT_EN: Enable the channel14 count 0 = DISABLE 1 = ENABLE
29	0x0	CH13_CNT_EN: Enable the channel13 count 0 = DISABLE 1 = ENABLE
28	0x0	CH12_CNT_EN: Enable the channel12 count 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
27	0x0	CH11_CNT_EN: Enable the channel11 count 0 = DISABLE 1 = ENABLE
26	0x0	CH10_CNT_EN: Enable the channel10 count 0 = DISABLE 1 = ENABLE
25	0x0	CH9_CNT_EN: Enable the channel9 count 0 = DISABLE 1 = ENABLE
24	0x0	CH8_CNT_EN: Enable the channel8 count 0 = DISABLE 1 = ENABLE
23	0x0	CH7_CNT_EN: Enable the channel7 count 0 = DISABLE 1 = ENABLE
22	0x0	CH6_CNT_EN: Enable the channel6 count 0 = DISABLE 1 = ENABLE
21	0x0	CH5_CNT_EN: Enable the channel5 count 0 = DISABLE 1 = ENABLE
20	0x0	CH4_CNT_EN: Enable the channel4 count 0 = DISABLE 1 = ENABLE
19	0x0	CH3_CNT_EN: Enable the channel3 count 0 = DISABLE 1 = ENABLE
18	0x0	CH2_CNT_EN: Enable the channel2 count 0 = DISABLE 1 = ENABLE
17	0x0	CH1_CNT_EN: Enable the channel1 count 0 = DISABLE 1 = ENABLE
16	0x0	CH0_CNT_EN: Enable the channel0 count 0 = DISABLE 1 = ENABLE
15:0	0x0	COUNT_VALUE: DMA COUNT Value.

11.2.3.6 APBDMA_IRQ_STA_CPU_0

Gathers all the after-masking CPU directed IRQ status bits.

APB-DMA CPU IRQ Status Register

Offset: 014h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxx

Bit	Reset	Description
15	X	CH15: Gathers all the after-masking CPU directed IRQ status bits from channel15 0 = DISABLE 1 = ENABLE
14	X	CH14: Gathers all the after-masking CPU directed IRQ status bits from channel14 0 = DISABLE 1 = ENABLE
13	X	CH13: Gathers all the after-masking CPU directed IRQ status bits from channel13 0 = DISABLE 1 = ENABLE
12	X	CH12: Gathers all the after-masking CPU directed IRQ status bits from channel12 0 = DISABLE 1 = ENABLE
11	X	CH11: Gathers all the after-masking CPU directed IRQ status bits from channel11 0 = DISABLE 1 = ENABLE
10	X	CH10: Gathers all the after-masking CPU directed IRQ status bits from channel10 0 = DISABLE 1 = ENABLE
9	X	CH9: Gathers all the after-masking CPU directed IRQ status bits from channel9 0 = DISABLE 1 = ENABLE
8	X	CH8: Gathers all the after-masking CPU directed IRQ status bits from channel8 0 = DISABLE 1 = ENABLE
7	X	CH7: Gathers all the after-masking CPU directed IRQ status bits from channel7 0 = DISABLE 1 = ENABLE
6	X	CH6: Gathers all the after-masking CPU directed IRQ status bits from channel6 0 = DISABLE 1 = ENABLE
5	X	CH5: Gathers all the after-masking CPU directed IRQ status bits from channel5 0 = DISABLE 1 = ENABLE
4	X	CH4: Gathers all the after-masking CPU directed IRQ status bits from channel4 0 = DISABLE 1 = ENABLE
3	X	CH3: Gathers all the after-masking CPU directed IRQ status bits from channel3 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	X	CH2: Gathers all the after-masking CPU directed IRQ status bits from channel2 0 = DISABLE 1 = ENABLE
1	X	CH1: Gathers all the after-masking CPU directed IRQ status bits from channel1 0 = DISABLE 1 = ENABLE
0	X	CH0: Gathers all the after-masking CPU directed IRQ status bits from channel0

11.2.3.7 APBDMA_IRQ_STA_COP_0

Gathers all the after-masking COP directed IRQ status bits.

APB-DMA COP IRQ Status Register

Offset: 018h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxx

Bit	Reset	Description
15	X	CH15: Gathers all the after-masking COP directed IRQ status bits from channel15 0 = DISABLE 1 = ENABLE
14	X	CH14: Gathers all the after-masking COP directed IRQ status bits from channel14 0 = DISABLE 1 = ENABLE
13	X	CH13: Gathers all the after-masking COP directed IRQ status bits from channel13 0 = DISABLE 1 = ENABLE
12	X	CH12: Gathers all the after-masking COP directed IRQ status bits from channel12 0 = DISABLE 1 = ENABLE
11	X	CH11: Gathers all the after-masking COP directed IRQ status bits from channel11 0 = DISABLE 1 = ENABLE
10	X	CH10: Gathers all the after-masking COP directed IRQ status bits from channel10 0 = DISABLE 1 = ENABLE
9	X	CH9: Gathers all the after-masking COP directed IRQ status bits from channel9 0 = DISABLE 1 = ENABLE
8	X	CH8: Gathers all the after-masking COP directed IRQ status bits from channel8 0 = DISABLE 1 = ENABLE
7	X	CH7: Gathers all the after-masking COP directed IRQ status bits from channel7 0 = DISABLE 1 = ENABLE
6	X	CH6: Gathers all the after-masking COP directed IRQ status bits from channel6 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
5	X	CH5: Gathers all the after-masking COP directed IRQ status bits from channel5 0 = DISABLE 1 = ENABLE
4	X	CH4: Gathers all the after-masking COP directed IRQ status bits from channel4 0 = DISABLE 1 = ENABLE
3	X	CH3: Gathers all the after-masking COP directed IRQ status bits from channel3 0 = DISABLE 1 = ENABLE
2	X	CH2: Gathers all the after-masking COP directed IRQ status bits from channel2 0 = DISABLE 1 = ENABLE
1	X	CH1: Gathers all the after-masking COP directed IRQ status bits from channel1 0 = DISABLE 1 = ENABLE
0	X	CH0: Gathers all the after-masking COP directed IRQ status bits from channel0

11.2.3.8 APBDMA_IRQ_MASK_0

Allows the IRQ to propagate when enabled, this is the ANDed result of set and clear.

APB-DMA IRQ Mask Register

Offset: 01ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxx

Bit	Reset	Description
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

Bit	Reset	Description
8	X	CH8: Each bit allows the associated channel8 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
7	X	CH7: Each bit allows the associated channel7 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
6	X	CH6: Each bit allows the associated channel6 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
5	X	CH5: Each bit allows the associated channel5 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
4	X	CH4: Each bit allows the associated channel4 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
3	X	CH3: Each bit allows the associated channel3 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
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'

11.2.3.9 APBDMA_IRQ_MASK_SET_0

APB-DMA IRQ Mask Set Register

Offset: 020h | Read/Write: WO | Reset: 0b00000000000000

Bit	Reset	Description
15	0x0	CH15: Sets the Mask Register 0 = DISABLE 1 = ENABLE
14	0x0	CH14: Sets the Mask Register 0 = DISABLE 1 = ENABLE
13	0x0	CH13: Sets the Mask Register 0 = DISABLE 1 = ENABLE
12	0x0	CH12: Sets the Mask Register 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
11	0x0	CH11: Sets the Mask Register 0 = DISABLE 1 = ENABLE
10	0x0	CH10: Sets the Mask Register 0 = DISABLE 1 = ENABLE
9	0x0	CH9: Sets the Mask Register 0 = DISABLE 1 = ENABLE
8	0x0	CH8: Sets the Mask Register 0 = DISABLE 1 = ENABLE
7	0x0	CH7: Sets the Mask Register 0 = DISABLE 1 = ENABLE
6	0x0	CH6: Sets the Mask Register 0 = DISABLE 1 = ENABLE
5	0x0	CH5: Sets the Mask Register 0 = DISABLE 1 = ENABLE
4	0x0	CH4: Sets the Mask Register 0 = DISABLE 1 = ENABLE
3	0x0	CH3: Sets the Mask Register 0 = DISABLE 1 = ENABLE
2	0x0	CH2: Sets the Mask Register 0 = DISABLE 1 = ENABLE
1	0x0	CH1: Sets the Mask Register 0 = DISABLE 1 = ENABLE
0	0x0	CH0: Sets the Mask Register 0 = DISABLE 1 = ENABLE

11.2.3.10 APBDMA_IRQ_MASK_CLR_0

APB-DMA IRQ Mask Clear Register

Offset: 024h | Read/Write: WO | Reset: 0b0000000000000000

Bit	Reset	Description
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

11.2.3.11 APBDMA_TRIG_REG_0

APB-DMA Trigger Register

Offset: 028h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
24	X	APB_15: EOC-15 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
23	X	APB_14: EOC-14 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
22	X	APB_13: EOC-13 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
21	X	APB_12: EOC-12 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
20	X	APB_11: EOC-11 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
19	X	APB_10: EOC-10 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
18	X	APB_9: EOC-9 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
17	X	APB_8: EOC-8 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
16	X	APB_7: EOC-7 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE

Bit	Reset	Description
15	X	APB_6: EOC-6 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
14	X	APB_5: EOC-5 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
13	X	APB_4: EOC-4 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
12	X	APB_3: EOC-3 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
11	X	APB_2: EOC-2 Initiated DMA Request after transfer completion) 0 = NOT_ACTIVE 1 = ACTIVE
10	X	APB_1: EOC-1 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
9	X	APB_0: EOC-0 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
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 SW initiated DMA request 0 = NOT_ACTIVE 1 = ACTIVE
3	X	SMP_26: Semaphore requests SW initiated DMA request 0 = NOT_ACTIVE 1 = ACTIVE
2	X	SMP_25: Semaphore requests SW initiated DMA request 0 = NOT_ACTIVE 1 = ACTIVE
1	X	SMP_24: Semaphore requests SW initiated DMA request 0 = NOT_ACTIVE 1 = ACTIVE

11.2.4 APB DMA Channel Registers

11.2.4.1 APBDMACHAN_CHANNEL_0_CSR_0

Writing a 1 to bit [31] of an APB DMA Channel Control Register will initiate the APB DMA Transfer. Because this action will depend on some values programmed in the other registers, it is recommended that the registers in the address space in the required APB DMA channel be programmed before writing into control register.

In "Once" mode, the channel is disabled after the APB DMA Transfer has completed. When the channel's transfer completes, an interrupt will be sent if the IE.EOC bit is set. If the transfer requires a trigger or flow, then the corresponding trigger selected by the "TRIG_SEL" or "REQ_SEL" field must become active respectively before the channel can starts its transfer. If both Trigger and Flow bits are set, both conditions must be met before the channel starts each DMA Burst.

APB-DMA-0 Control Register

Offset: 000h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: Enable DMA channel transfer 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: Interrupt when DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
29	0x0	HOLD: Hold this Processor until DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
28	0x0	DIR: DMA Transfer Direction 1 = AHB read to APB write 0 = APB read to AHB write 0 = AHB_WRITE 1 = AHB_READ
27	0x0	ONCE: Run Once or Run Multiple Mode (Allow Retriggering of this Channel) 1 = Run for One Block Transfer 0 = Run for Multiple Block Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK

Bit	Reset	Description
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 2 = SMP25 3 = SMP26 4 = SMP27 5 = XRQ_A 6 = XRQ_B 7 = TMR1 8 = TMR2 9 = APB_0 10 = APB_1 11 = APB_2 12 = APB_3 13 = APB_4 14 = APB_5 15 = APB_6 16 = APB_7 17 = APB_8 18 = APB_9 19 = APB_10 20 = APB_11 21 = APB_12 22 = APB_13 23 = APB_14 24 = APB_15
21	0x0	FLOW: Flow Control Enable (Synchronize Burst Transfers) 1 = Link to DRQ source 0 = Independent of DRQ request 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = I2S_2 2 = I2S_1 3 = SPD_I 4 = UI_I 5 = MIPI 6 = I2S2_2 7 = I2S2_1 8 = UART1 9 = UART2 10 = UART3 11 = SPI 12 = AC97 13 = AC Modem 14 = RSVD 15 = SPI1 16 = SPI2 17 = SPI3 18 = SPI4 19 = UART4 20 = UART5 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = NA26 27 = NA27 28 = NA28 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

11.2.4.2 APBDMACHAN_CHANNEL_0_STA_0

APB-DMA-0 Status Register

Offset: 004h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicates whether DMA Channel Status active or not 0 = WAIT 1 = ACTIVE
30	RW	0x0	ISE_EOC: Write '1' to clear the flag 0 = NO_INTR 1 = INTR
29	RO	X	HALT: Holding Status of Processor 0 = NO_HALT 1 = HALT

Bit	R/W	Reset	Description
28	RO	X	PING_PONG_STA: if dir = AHB_WRITE : 0 - ping buffer transfer completed ; 1 - pong buffer transfer completed if dir = AHB_READ : 1 - ping buffer transfer completed ; 0 - pong buffer transfer completed 0 = PING_INTR_STS 1 = PONG_INTR_STS
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

11.2.4.3 APBDMACHAN_CHANNEL_0_AHB_PTR_0

APB-DMA-0 AHB Starting Address Pointer Register

Offset: 010h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

11.2.4.4 APBDMACHAN_CHANNEL_0_AHB_SEQ_0

APB-DMA-0 AHB Address Sequencer Register

Offset: 014h | Read/Write: R/W | Reset: 0b00100000xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP
30:28	0x2	AHB_BUS_WIDTH: AHB Bus Width 0 = 8 bit Bus (RSVD) 1 = 16 bit Bus (RSVD) 2 = 32 bit Bus (Def) 3 = 64 bit Bus (RSVD) 4 = 128 bit Bus (RSVD) 0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	AHB_DATA_SWAP: When enabled the data going to AHB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24] }. 0 = DISABLE 1 = ENABLE
26:24	0x0	AHB_BURST: AHB Burst Size DMA Burst Length (encoded) 4 = 1 Word (1x32bits) 5 = 4 Words (4x32bits) else = 8 Words (8x32bits) default 4 = DMA_BURST_1WORDS 5 = DMA_BURST_4WORDS 6 = DMA_BURST_8WORDS
19	0x0	DBL_BUF: 2X Double Buffering Mode (For Run-Multiple Mode with No Wrap Operations) 1 = Reload Base Address for 2X blocks (reload every other time) 0 = Reload Base Address for 1X blocks (def) (reload each time) 0 = RELOAD_FOR_1X_BLOCKS 1 = RELOAD_FOR_2X_BLOCKS

Bit	Reset	Description
18:16	0x0	WRAP: AHB Address Wrap: AHB Address wrap-around window 0=No Wrap (default) 5=Wrap on 512 word window 1=Wrap on 32 word window 6=Wrap on 1024 word window 2=Wrap on 64 word window 7=Wrap on 2048 word window 3=Wrap on 128 word window 4=Wrap on 256 word window 0 = NO_WRAP 1 = WRAP_0N_32WORDS 2 = WRAP_ON_64WORDS 3 = WRAP_ON_128WORDS 4 = WRAP_ON_256WORDS 5 = WRAP_ON_512WORDS 6 = WRAP_ON_1024WORDS 7 = WRAP_ON_2048WORDS

11.2.4.5 APBDMACHAN_CHANNEL_0_APB_PTR_0

APB-DMA-0 APB Starting Address Pointer Register

Offset: 018h | Read/Write: R/W | Reset: 0b00000000000000

Bit	Reset	Description
15:2	0x0	APB_BASE: APB-DMA Starting address for APB Bus: APB Base address: Upper 16 bits are fixed at 0x7000:XXXX

11.2.4.6 APBDMACHAN_CHANNEL_0_APB_SEQ_0

APB-DMA-0 APB Address Sequencer Assignments

Offset: 01ch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 3 = 64 bit Bus (RSVD) 4 = 128 bit BUS (RSVD) 0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	APB_DATA_SWAP: When enabled the data going to APB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24]}. 0 = DISBALE 1 = ENABLE

Bit	Reset	Description
18:16	0x1	APB_ADDR_WRAP: APB Address Wrap-around Window 0 = No Wrap 1 = Wrap on 1 Word Window (def) 2 = Wrap on 2 Word Window 3 = Wrap on 4 Word Window 4 = Wrap on 8 Word Window 5 = Wrap on 16 Word Window 6 = Wrap on 32 Word Window 7 = Wrap on 64 Word Window (RSVD) 0 = NO_WRAP 1 = WRAP_0N_1WORDS 2 = WRAP_ON_2WORDS 3 = WRAP_ON_4WORDS 4 = WRAP_ON_8WORDS 5 = WRAP_ON_16WORDS 6 = WRAP_ON_32WORDS 7 = WRAP_ON_64WORDS

11.2.4.7 APBDMACHAN_CHANNEL_1_CSR_0

APB-DMA-1 Control

Offset: 020h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: Enable DMA channel transfer 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: Interrupt when DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
29	0x0	HOLD: Hold this Processor until DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
28	0x0	DIR: DMA Transfer Direction 1 = AHB read to APB write 0 = APB read to AHB write 0 = AHB_WRITE 1 = AHB_READ
27	0x0	ONCE: Run Once or Run Multiple Mode (Allow Retriggering of this Channel) 1 = Run for One Block Transfer 0 = Run for Multiple Block Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK

Bit	Reset	Description
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 2 = SMP25 3 = SMP26 4 = SMP27 5 = XRQ_A 6 = XRQ_B 7 = TMR1 8 = TMR2 9 = APB_0 10 = APB_1 11 = APB_2 12 = APB_3 13 = APB_4 14 = APB_5 15 = APB_6 16 = APB_7 17 = APB_8 18 = APB_9 19 = APB_10 20 = APB_11 21 = APB_12 22 = APB_13 23 = APB_14 24 = APB_15
21	0x0	FLOW: Flow Control Enable (Synchronize Burst Transfers) 1 = Link to DRQ source 0 = Independent of DRQ request 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = I2S_2 2 = I2S_1 3 = SPD_I 4 = UI_I 5 = MIPI 6 = I2S2_2 7 = I2S2_1 8 = UART1 9 = UART2 10 = UART3 11 = SPI 12 = AC97 13 = AC Modem 14 = RSVD 15 = SPI1 16 = SPI2 17 = SPI3 18 = SL2B4 19 = UART4 20 = UART5 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = NA26 27 = NA27 28 = NA28 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

11.2.4.8 APBDMACHAN_CHANNEL_1_STA_0

APB-DMA-1 Status Register

Offset: 024h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicates whether DMA Channel Status active or not 0 = WAIT 1 = ACTIVE
30	RW	0x0	ISE_EOC: Write '1' to clear the flag 0 = NO_INTR 1 = INTR
29	RO	X	HALT: Holding Status of Processor 0 = NO_HALT 1 = HALT

Bit	R/W	Reset	Description
28	RO	X	PING_PONG_STS: 0 = PING_INTR_STS 1 = PONG_INTR_STS
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

11.2.4.9 APBDMACHAN_CHANNEL_1_AHB_PTR_0

APB-DMA-1 AHB Starting Address Pointer Register

Offset: 030h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

11.2.4.10 APBDMACHAN_CHANNEL_1_AHB_SEQ_0

APB-DMA-1 AHB Address Sequencer Register

Offset: 034h | Read/Write: R/W | Reset: 0b00100000xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 1 = CPU 0 = COP
30:28	0x2	AHB_BUS_WIDTH: AHB Bus Width 0 = 8 bit Bus (RSVD) 1 = 16 bit Bus (RSVD) 2 = 32 bit Bus (Def) 3 = 64 bit Bus (RSVD)4 = 128 bit Bus (RSVD) 0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	AHB_DATA_SWAP: When enabled the data going to AHB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24]} 0 = DISABLE 1 = ENABLE
26:24	0x0	AHB_BURST: AHB Burst Size DMA Burst Length (encoded) 4 = 1 Word (1x32bits) 5 = 4 Words (4x32bits)else = 8 Words (8x32bits) default 4 = DMA_BURST_1WORDS 5 = DMA_BURST_4WORDS 6 = DMA_BURST_8WORDS
19	0x0	DBL_BUF: 2X Double Buffering Mode (For Run-Multiple Mode with No Wrap Operations) 1 = Reload Base Address for 2X blocks (reload every other time) 0 = Reload Base Address for 1X blocks (def) (reload each time) 0 = RELOAD_FOR_1X_BLOCKS 1 = RELOAD_FOR_2X_BLOCKS

Bit	Reset	Description
18:16	0x0	WRAP: AHB Address Wrap: AHB Address wrap-around window 0=No Wrap (default) 5=Wrap on 512 word window 1=Wrap on 32 word window 6=Wrap on 1024 word window 2=Wrap on 64 word window 7=Wrap on 2048 word window 3=Wrap on 128 word window 4=Wrap on 256 word window 0 = NO_WRAP 1 = WRAP_0N_32WORDS 2 = WRAP_ON_64WORDS 3 = WRAP_ON_128WORDS 4 = WRAP_ON_256WORDS 5 = WRAP_ON_512WORDS 6 = WRAP_ON_1024WORDS 7 = WRAP_ON_2048WORDS

11.2.4.11 APBDMACHAN_CHANNEL_1_APB_PTR_0

APB-DMA-1 APB Starting Address Pointer Register

Offset: 038h | Read/Write: R/W | Reset: 0b00000000000000

Bit	Reset	Description
15:2	0x0	APB_BASE: APB-DMA Starting address for APB Bus: APB Base address: Upper 16 bits are fixed at 0x7000:XXXX

11.2.4.12 APBDMACHAN_CHANNEL_1_APB_SEQ_0

APB-DMA-1 APB Address Sequencer Assignments

Offset: 03ch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 3 = 64 bit Bus (RSVD) 4 = 128 bit BUS (RSVD) 0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	APB_DATA_SWAP: When enabled the data going to APB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24]} 0 = DISBALE 1 = ENABLE

Bit	Reset	Description
18:16	0x1	APB_ADDR_WRAP: APB Address Wrap-around Window 0 = No Wrap 1 = Wrap on 1 Word Window (def) 2 = Wrap on 2 Word Window 3 = Wrap on 4 Word Window 4 = Wrap on 8 Word Window 5 = Wrap on 16 Word Window 6 = Wrap on 32 Word Window 7 = Wrap on 64 Word Window (RSVD) 0 = NO_WRAP 1 = WRAP_0N_1WORDS 2 = WRAP_ON_2WORDS 3 = WRAP_ON_4WORDS 4 = WRAP_ON_8WORDS 5 = WRAP_ON_16WORDS 6 = WRAP_ON_32WORDS 7 = WRAP_ON_64WORDS

11.2.4.13 APBDMACHAN_CHANNEL_2_CSR_0

APB-DMA-2 Control

Offset: 040h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: Enable DMA channel transfer 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: Interrupt when DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
29	0x0	HOLD: Hold this Processor until DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
28	0x0	DIR: DMA Transfer Direction 1 = AHB read to APB write 0 = APB read to AHB write 0 = AHB_WRITE 1 = AHB_READ
27	0x0	ONCE: Run Once or Run Multiple Mode (Allow Retriggering of this Channel) 1 = Run for One Block Transfer 0 = Run for Multiple Block Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK

Bit	Reset	Description
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 2 = SMP25 3 = SMP26 4 = SMP27 5 = XRQ_A 6 = XRQ_B 7 = TMR1 8 = TMR2 9 = APB_0 10 = APB_1 11 = APB_2 12 = APB_3 13 = APB_4 14 = APB_5 15 = APB_6 16 = APB_7 17 = APB_8 18 = APB_9 19 = APB_10 20 = APB_11 21 = APB_12 22 = APB_13 23 = APB_14 24 = APB_15
21	0x0	FLOW: Flow Control Enable (Synchronize Burst Transfers) 1 = Link to DRQ source 0 = Independent of DRQ request 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = I2S_2 2 = I2S_1 3 = SPD_I 4 = UI_I 5 = MIPI 6 = I2S2_2 7 = I2S2_1 8 = UART1 9 = UART2 10 = UART3 11 = SPI 12 = AC97 13 = AC Modem 14 = RSVD 15 = SPI1 16 = SPI2 17 = SPI3 18 = SPI4 19 = UART4 20 = UART5 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = NA26 27 = NA27 28 = NA28 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

11.2.4.14 APBDMACHAN_CHANNEL_2_STA_0

APB-DMA-2 Status Register

Offset: 044h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicates whether DMA Channel Status active or not 0 = WAIT 1 = ACTIVE
30	RW	0x0	ISE_EOC: Write '1' to clear the flag 0 = NO_INTR 1 = INTR
29	RO	X	HALT: Holding Status of Processor 0 = NO_HALT 1 = HALT

Bit	R/W	Reset	Description
28	RO	X	PING_PONG_STS: 0 = PING_INTR_STS 1 = PONG_INTR_STS
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

11.2.4.15 APBDMACHAN_CHANNEL_2_AHB_PTR_0

APB-DMA-2 AHB Starting Address Pointer Register

Offset: 050h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

11.2.4.16 APBDMACHAN_CHANNEL_2_AHB_SEQ_0

APB-DMA-2 AHB Address Sequencer Register

Offset: 054h | Read/Write: R/W | Reset: 0b00100000xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 1 = CPU 0 = COP
30:28	0x2	AHB_BUS_WIDTH: AHB Bus Width 0 = 8 bit Bus (RSVD) 1 = 16 bit Bus (RSVD) 2 = 32 bit Bus (Def) 3 = 64 bit Bus (RSVD)4 = 128 bit Bus (RSVD) 0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	AHB_DATA_SWAP: When enabled the data going to AHB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24]} 0 = DISABLE 1 = ENABLE
26:24	0x0	AHB_BURST: AHB Burst Size DMA Burst Length (encoded) 4 = 1 Word (1x32bits) 5 = 4 Words (4x32bits)else = 8 Words (8x32bits) default 4 = DMA_BURST_1WORDS 5 = DMA_BURST_4WORDS 6 = DMA_BURST_8WORDS
19	0x0	DBL_BUF: 2X Double Buffering Mode (For Run-Multiple Mode with No Wrap Operations) 1 = Reload Base Address for 2X blocks (reload every other time) 0 = Reload Base Address for 1X blocks (def) (reload each time) 0 = RELOAD_FOR_1X_BLOCKS 1 = RELOAD_FOR_2X_BLOCKS

Bit	Reset	Description
18:16	0x0	WRAP: AHB Address Wrap: AHB Address wrap-around window 0=No Wrap (default) 5=Wrap on 512 word window 1=Wrap on 32 word window 6=Wrap on 1024 word window 2=Wrap on 64 word window 7=Wrap on 2048 word window 3=Wrap on 128 word window 4=Wrap on 256 word window 0 = NO_WRAP 1 = WRAP_0N_32WORDS 2 = WRAP_ON_64WORDS 3 = WRAP_ON_128WORDS 4 = WRAP_ON_256WORDS 5 = WRAP_ON_512WORDS 6 = WRAP_ON_1024WORDS 7 = WRAP_ON_2048WORDS

11.2.4.17 APBDMACHAN_CHANNEL_2_APB_PTR_0

APB-DMA-2 APB Starting Address Pointer Register

Offset: 058h | Read/Write: R/W | Reset: 0b00000000000000

Bit	Reset	Description
15:2	0x0	APB_BASE: APB-DMA Starting address for APB Bus: APB Base address: Upper 16 bits are fixed at 0x7000:XXXX

11.2.4.18 APBDMACHAN_CHANNEL_2_APB_SEQ_0

APB-DMA-2 APB Address Sequencer Assignments

Offset: 05ch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 3 = 64 bit Bus (RSVD) 4 = 128 bit BUS (RSVD) 0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	APB_DATA_SWAP: When enabled the data going to APB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24]} 0 = DISBALE 1 = ENABLE

Bit	Reset	Description
18:16	0x1	APB_ADDR_WRAP: APB Address Wrap-around Window 0 = No Wrap 1 = Wrap on 1 Word Window (def) 2 = Wrap on 2 Word Window 3 = Wrap on 4 Word Window 4 = Wrap on 8 Word Window 5 = Wrap on 16 Word Window 6 = Wrap on 32 Word Window 7 = Wrap on 64 Word Window (RSVD) 0 = NO_WRAP 1 = WRAP_0N_1WORDS 2 = WRAP_ON_2WORDS 3 = WRAP_ON_4WORDS 4 = WRAP_ON_8WORDS 5 = WRAP_ON_16WORDS 6 = WRAP_ON_32WORDS 7 = WRAP_ON_64WORDS

11.2.4.19 APBDMACHAN_CHANNEL_3_CSR_0

APB-DMA-3 Control

Offset: 060h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: Enable DMA channel transfer 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: Interrupt when DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
29	0x0	HOLD: Hold this Processor until DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
28	0x0	DIR: DMA Transfer Direction 1 = AHB read to APB write 0 = APB read to AHB write 0 = AHB_WRITE 1 = AHB_READ
27	0x0	ONCE: Run Once or Run Multiple Mode (Allow Retriggering of this Channel) 1 = Run for One Block Transfer 0 = Run for Multiple Block Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK

Bit	Reset	Description
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 2 = SMP25 3 = SMP26 4 = SMP27 5 = XRQ_A 6 = XRQ_B 7 = TMR1 8 = TMR2 9 = APB_0 10 = APB_1 11 = APB_2 12 = APB_3 13 = APB_4 14 = APB_5 15 = APB_6 16 = APB_7 17 = APB_8 18 = APB_9 19 = APB_10 20 = APB_11 21 = APB_12 22 = APB_13 23 = APB_14 24 = APB_15
21	0x0	FLOW: Flow Control Enable (Synchronize Burst Transfers) 1 = Link to DRQ source 0 = Independent of DRQ request 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = I2S_2 2 = I2S_1 3 = SPD_I 4 = UI_I 5 = MIPI 6 = I2S2_2 7 = I2S2_1 8 = UART1 9 = UART2 10 = UART3 11 = SPI 12 = AC97 13 = AC Modem 14 = RSVD 15 = SPI1 16 = SPI2 17 = SPI3 18 = SPI4 19 = UART4 20 = UART5 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR26 = NA26 27 = NA27 28 = NA28 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

11.2.4.20 APBDMACHAN_CHANNEL_3_STA_0

APB-DMA-3 Status Register

Offset: 064h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicates whether DMA Channel Status active 0 = WAIT 1 = ACTIVE
30	RW	0x0	ISE_EOC: Write '1' to clear the flag 0 = NO_INTR 1 = INTR
29	RO	X	HALT: Holding Status of Processor 0 = NO_HALT 1 = HALT
28	RO	X	PING_PONG_STS: 0 = PING_INTR_STS 1 = PONG_INTR_STS

Bit	R/W	Reset	Description
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

11.2.4.21 APBDMACHAN_CHANNEL_3_AHB_PTR_0

APB-DMA-3 AHB Starting Address Pointer Register

Offset: 070h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

11.2.4.22 APBDMACHAN_CHANNEL_3_AHB_SEQ_0

APB-DMA-3 AHB Address Sequencer Register

Offset: 074h | Read/Write: R/W | Reset: 0b00100000xxxxx000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 1 = CPU 0 = COP
30:28	0x2	AHB_BUS_WIDTH: AHB Bus Width 0 = 8 bit Bus (RSVD) 1 = 16 bit Bus (RSVD) 2 = 32 bit Bus (Def) 3 = 64 bit Bus (RSVD)4 = 128 bit Bus (RSVD) 0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	AHB_DATA_SWAP: When enabled the data going to AHB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24]} 0 = DISABLE 1 = ENABLE
26:24	0x0	AHB_BURST: AHB Burst Size DMA Burst Length (encoded) 4 = 1 Word (1x32bits) 5 = 4 Words (4x32bits)else = 8 Words (8x32bits) default 4 = DMA_BURST_1WORDS 5 = DMA_BURST_4WORDS 6 = DMA_BURST_8WORDS
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

11.2.4.23 APBDMACHAN_CHANNEL_3_APB_PTR_0

APB-DMA-3 APB Starting Address Pointer Register

Offset: 078h | Read/Write: R/W | Reset: 0b00000000000000

Bit	Reset	Description
15:2	0x0	APB_BASE: APB-DMA Starting address for APB Bus: APB Base address: Upper 16 bits are fixed at 0x7000:XXXX

11.2.4.24 APBDMACHAN_CHANNEL_3_APB_SEQ_0

APB-DMA-3 APB Address Sequencer Assignments

Offset: 07ch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 3 = 64 bit Bus (RSVD) 4 = 128 bit BUS (RSVD) 0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	APB_DATA_SWAP: When enabled the data going to APB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24] } 0 = DISBALE 1 = ENABLE
18:16	0x1	APB_ADDR_WRAP: APB Address Wrap-around Window 0 = No Wrap 1 = Wrap on 1 Word Window (def) 2 = Wrap on 2 Word Window 3 = Wrap on 4 Word Window 4 = Wrap on 8 Word Window 5 = Wrap on 16 Word Window 6 = Wrap on 32 Word Window 7 = Wrap on 64 Word Window (RSVD) 0 = NO_WRAP 1 = WRAP_0N_1WORDS 2 = WRAP_ON_2WORDS 3 = WRAP_ON_4WORDS 4 = WRAP_ON_8WORDS 5 = WRAP_ON_16WORDS 6 = WRAP_ON_32WORDS 7 = WRAP_ON_64WORDS

11.2.4.25 APBDMACHAN_CHANNEL_4_CSR_0

APB-DMA-4 Control

Offset: 080h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: Enable DMA channel transfer 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
30	0x0	IE_EOC: Interrupt when DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
29	0x0	HOLD: Hold this Processor until DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
28	0x0	DIR: DMA Transfer Direction 1 = AHB read to APB write 0 = APB read to AHB write 0 = AHB_WRITE 1 = AHB_READ
27	0x0	ONCE: Run Once or Run Multiple Mode (Allow Retriggering of this Channel) 1 = Run for One Block Transfer 0 = Run for Multiple Block Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 2 = SMP25 3 = SMP26 4 = SMP27 5 = XRQ_A 6 = XRQ_B 7 = TMR1 8 = TMR2 9 = APB_0 10 = APB_1 11 = APB_2 12 = APB_3 13 = APB_4 14 = APB_5 15 = APB_6 16 = APB_7 17 = APB_8 18 = APB_9 19 = APB_10 20 = APB_11 21 = APB_12 22 = APB_13 23 = APB_14 24 = APB_15
21	0x0	FLOW: Flow Control Enable (Synchronize Burst Transfers) 1 = Link to DRQ source 0 = Independent of DRQ request 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = I2S_2 2 = I2S_1 3 = SPD_I 4 = UI_I 5 = MIPI 6 = I2S2_2 7 = I2S2_1 8 = UART1 9 = UART2 10 = UART3 11 = SPI 12 = AC97 13 = AC Modem 14 = RSVD 15 = SPI1 16 = SPI2 17 = SPI3 18 = SPI4 19 = UART4 20 = UART5 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = NA26 27 = NA27 28 = NA28 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

11.2.4.26 APBDMACHAN_CHANNEL_4_STA_0

APB-DMA-4 Status Register

Offset: 084h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicates whether DMA Channel Status active or not 0 = WAIT 1 = ACTIVE
30	RW	0x0	ISE_EOC: Write '1' to clear the flag 0 = NO_INTR 1 = INTR
29	RO	X	HALT: Holding Status of Processor 0 = NO_HALT 1 = HALT

Bit	R/W	Reset	Description
28	RO	X	PING_PONG_STS: 0 = PING_INTR_STS 1 = PONG_INTR_STS
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

11.2.4.27 APBDMACHAN_CHANNEL_4_AHB_PTR_0

APB-DMA-4 AHB Starting Address Pointer Register

Offset: 090h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

11.2.4.28 APBDMACHAN_CHANNEL_4_AHB_SEQ_0

APB-DMA-4 AHB Address Sequencer Register

Offset: 094h | Read/Write: R/W | Reset: 0b00100000xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 1 = CPU 0 = COP
30:28	0x2	AHB_BUS_WIDTH: AHB Bus Width 0 = 8 bit Bus (RSVD) 1 = 16 bit Bus (RSVD) 2 = 32 bit Bus (Def) 3 = 64 bit Bus (RSVD) 4 = 128 bit Bus (RSVD) 0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	AHB_DATA_SWAP: When enabled the data going to AHB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24] } 0 = DISABLE 1 = ENABLE
26:24	0x0	AHB_BURST: AHB Burst Size DMA Burst Length (encoded) 4 = 1 Word (1x32bits) 5 = 4 Words (4x32bits)else = 8 Words (8x32bits) default 4 = DMA_BURST_1WORDS 5 = DMA_BURST_4WORDS 6 = DMA_BURST_8WORDS
19	0x0	DBL_BUF: 2X Double Buffering Mode (For Run-Multiple Mode with No Wrap Operations) 1 = Reload Base Address for 2X blocks (reload every other time) 0 = Reload Base Address for 1X blocks (def) (reload each time) 0 = RELOAD_FOR_1X_BLOCKS 1 = RELOAD_FOR_2X_BLOCKS

Bit	Reset	Description
18:16	0x0	WRAP: AHB Address Wrap: AHB Address wrap-around window 0=No Wrap (default) 5=Wrap on 512 word window 1=Wrap on 32 word window 6=Wrap on 1024 word window 2=Wrap on 64 word window 7=Wrap on 2048 word window 3=Wrap on 128 word window 4=Wrap on 256 word window 0 = NO_WRAP 1 = WRAP_0N_32WORDS 2 = WRAP_ON_64WORDS 3 = WRAP_ON_128WORDS 4 = WRAP_ON_256WORDS 5 = WRAP_ON_512WORDS 6 = WRAP_ON_1024WORDS 7 = WRAP_ON_2048WORDS

11.2.4.29 APBDMACHAN_CHANNEL_4_APB_PTR_0

APB-DMA-4 APB Starting Address Pointer Register

Offset: 098h | Read/Write: R/W | Reset: 0b00000000000000

Bit	Reset	Description
15:2	0x0	APB_BASE: APB-DMA Starting address for APB Bus: APB Base address: Upper 16 bits are fixed at 0x7000:XXXX

11.2.4.30 APBDMACHAN_CHANNEL_4_APB_SEQ_0

APB-DMA-4 APB Address Sequencer Assignments

Offset: 09ch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 3 = 64 bit Bus (RSVD) 4 = 128 bit BUS (RSVD) 0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	APB_DATA_SWAP: When enabled the data going to APB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24]} 0 = DISBALE 1 = ENABLE

Bit	Reset	Description
18:16	0x1	APB_ADDR_WRAP: APB Address Wrap-around Window 0 = No Wrap 1 = Wrap on 1 Word Window (def) 2 = Wrap on 2 Word Window 3 = Wrap on 4 Word Window 4 = Wrap on 8 Word Window 5 = Wrap on 16 Word Window 6 = Wrap on 32 Word Window 7 = Wrap on 64 Word Window (RSVD) 0 = NO_WRAP 1 = WRAP_0N_1WORDS 2 = WRAP_ON_2WORDS 3 = WRAP_ON_4WORDS 4 = WRAP_ON_8WORDS 5 = WRAP_ON_16WORDS 6 = WRAP_ON_32WORDS 7 = WRAP_ON_64WORDS

11.2.4.31 APBDMACHAN_CHANNEL_5_CSR_0

APB-DMA-5 Control

Offset: 0a0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: Enable DMA channel transfer 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: Interrupt when DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
29	0x0	HOLD: Hold this Processor until DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
28	0x0	DIR: DMA Transfer Direction 1 = AHB read to APB write 0 = APB read to AHB write 0 = AHB_WRITE 1 = AHB_READ
27	0x0	ONCE: Run Once or Run Multiple Mode (Allow Retriggering of this Channel) 1 = Run for One Block Transfer 0 = Run for Multiple Block Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK

Bit	Reset	Description
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 2 = SMP25 3 = SMP26 4 = SMP27 5 = XRQ_A 6 = XRQ_B 7 = TMR1 8 = TMR2 9 = APB_0 10 = APB_1 11 = APB_2 12 = APB_3 13 = APB_4 14 = APB_5 15 = APB_6 16 = APB_7 17 = APB_8 18 = APB_9 19 = APB_10 20 = APB_11 21 = APB_12 22 = APB_13 23 = APB_14 24 = APB_15
21	0x0	FLOW: Flow Control Enable (Synchronize Burst Transfers) 1 = Link to DRQ source 0 = Independent of DRQ request 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = I2S_2 2 = I2S_1 3 = SPD_I 4 = UI_I 5 = MIPI 6 = I2S2_2 7 = I2S2_1 8 = UART1 9 = UART2 10 = UART3 11 = SPI 12 = AC97 13 = AC Modem 14 = RSVD 15 = SPI1 16 = SPI2 17 = SPI3 18 = SPI4 19 = UART4 20 = UART5 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = NA26 27 = NA27 28 = NA28 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

11.2.4.32 APBDMACHAN_CHANNEL_5_STA_0

APB-DMA-5 Status Register

Offset: 0a4h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicate whether DMA Channel Status active or not 0 = WAIT 1 = ACTIVE
30	RW	0x0	ISE_EOC: Write '1' to clear the flag 0 = NO_INTR 1 = INTR
29	RO	X	HALT: Holding Status of Processor 0 = NO_HALT 1 = HALT

Bit	R/W	Reset	Description
28	RO	X	PING_PONG_STS: 0 = PING_INTR_STS 1 = PONG_INTR_STS
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

11.2.4.33 APBDMACHAN_CHANNEL_5_AHB_PTR_0

APB-DMA-5 AHB Starting Address Pointer Register

Offset: 0b0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

11.2.4.34 APBDMACHAN_CHANNEL_5_AHB_SEQ_0

APB-DMA-5 AHB Address Sequencer Register

Offset: 0b4h | Read/Write: R/W | Reset: 0b00100000xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 1 = CPU 0 = COP
30:28	0x2	AHB_BUS_WIDTH: AHB Bus Width 0 = 8 bit Bus (RSVD) 1 = 16 bit Bus (RSVD) 2 = 32 bit Bus (Def) 3 = 64 bit Bus (RSVD)4 = 128 bit Bus (RSVD) 0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	AHB_DATA_SWAP: When enabled the data going to AHB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24]} 0 = DISABLE 1 = ENABLE
26:24	0x0	AHB_BURST: AHB Burst Size DMA Burst Length (encoded) 4 = 1 Word (1x32bits) 5 = 4 Words (4x32bits)else = 8 Words (8x32bits) default 4 = DMA_BURST_1WORDS 5 = DMA_BURST_4WORDS 6 = DMA_BURST_8WORDS
19	0x0	DBL_BUF: 2X Double Buffering Mode (For Run-Multiple Mode with No Wrap Operations) 1 = Reload Base Address for 2X blocks (reload every other time) 0 = Reload Base Address for 1X blocks (def) (reload each time) 0 = RELOAD_FOR_1X_BLOCKS 1 = RELOAD_FOR_2X_BLOCKS

Bit	Reset	Description
18:16	0x0	WRAP: AHB Address Wrap: AHB Address wrap-around window 0=No Wrap (default) 5=Wrap on 512 word window 1=Wrap on 32 word window 6=Wrap on 1024 word window 2=Wrap on 64 word window 7=Wrap on 2048 word window 3=Wrap on 128 word window 4=Wrap on 256 word window 0 = NO_WRAP 1 = WRAP_0N_32WORDS 2 = WRAP_ON_64WORDS 3 = WRAP_ON_128WORDS 4 = WRAP_ON_256WORDS 5 = WRAP_ON_512WORDS 6 = WRAP_ON_1024WORDS 7 = WRAP_ON_2048WORDS

11.2.4.35 APBDMACHAN_CHANNEL_5_APB_PTR_0

APB-DMA-5 APB Starting Address Pointer Register

Offset: 0b8h | Read/Write: R/W | Reset: 0b00000000000000

Bit	Reset	Description
15:2	0x0	APB_BASE: APB-DMA Starting address for APB Bus: APB Base address: Upper 16 bits are fixed at 0x7000:XXXX

11.2.4.36 APBDMACHAN_CHANNEL_5_APB_SEQ_0

APB-DMA-5 APB Address Sequencer Assignments

Offset: 0bch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 3 = 64 bit Bus (RSVD) 4 = 128 bit BUS (RSVD) 0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	APB_DATA_SWAP: When enabled the data going to APB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24]} 0 = DISBALE 1 = ENABLE

Bit	Reset	Description
18:16	0x1	APB_ADDR_WRAP: APB Address Wrap-around Window 0 = No Wrap 1 = Wrap on 1 Word Window (def) 2 = Wrap on 2 Word Window 3 = Wrap on 4 Word Window 4 = Wrap on 8 Word Window 5 = Wrap on 16 Word Window 6 = Wrap on 32 Word Window 7 = Wrap on 64 Word Window (RSVD) 0 = NO_WRAP 1 = WRAP_0N_1WORDS 2 = WRAP_ON_2WORDS 3 = WRAP_ON_4WORDS 4 = WRAP_ON_8WORDS 5 = WRAP_ON_16WORDS 6 = WRAP_ON_32WORDS 7 = WRAP_ON_64WORDS

11.2.4.37 APBDMACHAN_CHANNEL_6_CSR_0

APB-DMA-6 Control

Offset: 0c0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: Enable DMA channel transfer 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: Interrupt when DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
29	0x0	HOLD: Hold this Processor until DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
28	0x0	DIR: DMA Transfer Direction 1 = AHB read to APB write 0 = APB read to AHB write 0 = AHB_WRITE 1 = AHB_READ
27	0x0	ONCE: Run Once or Run Multiple Mode (Allow Retriggering of this Channel) 1 = Run for One Block Transfer 0 = Run for Multiple Block Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK

Bit	Reset	Description
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 2 = SMP25 3 = SMP26 4 = SMP27 5 = XRQ_A 6 = XRQ_B 7 = TMR1 8 = TMR2 9 = APB_0 10 = APB_1 11 = APB_2 12 = APB_3 13 = APB_4 14 = APB_5 15 = APB_6 16 = APB_7 17 = APB_8 18 = APB_9 19 = APB_10 20 = APB_11 21 = APB_12 22 = APB_13 23 = APB_14 24 = APB_15
21	0x0	FLOW: Flow Control Enable (Synchronize Burst Transfers) 1 = Link to DRQ source 0 = Independent of DRQ request 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = I2S_2 2 = I2S_1 3 = SPD_I 4 = UI_I 5 = MIPI 6 = I2S2_2 7 = I2S2_1 8 = UART1 9 = UART2 10 = UART3 11 = SPI 12 = AC97 13 = AC Modem 14 = RSVD 15 = SPI1 16 = SPI2 17 = SPI3 18 = SPI4 19 = UART4 20 = UART5 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = NA26 27 = NA27 28 = NA28 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

11.2.4.38 APBDMACHAN_CHANNEL_6_STA_0

APB-DMA-6 Status Register

Offset: 0c4h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicate whether DMA Channel Status active or not 0 = WAIT 1 = ACTIVE
30	RW	0x0	ISE_EOC: Write '1' to clear the flag 0 = NO_INTR 1 = INTR
29	RO	X	HALT: Holding Status of Processor 0 = NO_HALT 1 = HALT

Bit	R/W	Reset	Description
28	RO	X	PING_PONG_STS: 0 = PING_INTR_STS 1 = PONG_INTR_STS
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

11.2.4.39 APBDMACHAN_CHANNEL_6_AHB_PTR_0

APB-DMA-6 AHB Starting Address Pointer Register

Offset: 0d0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

11.2.4.40 APBDMACHAN_CHANNEL_6_AHB_SEQ_0

APB-DMA-6 AHB Address Sequencer Register

Offset: 0d4h | Read/Write: R/W | Reset: 0b00100000xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 1 = CPU 0 = COP
30:28	0x2	AHB_BUS_WIDTH: AHB Bus Width 0 = 8 bit Bus (RSVD) 1 = 16 bit Bus (RSVD) 2 = 32 bit Bus (Def) 3 = 64 bit Bus (RSVD)4 = 128 bit Bus (RSVD) 0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	AHB_DATA_SWAP: When enabled the data going to AHB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24]} 0 = DISABLE 1 = ENABLE
26:24	0x0	AHB_BURST: AHB Burst Size DMA Burst Length (encoded) 4 = 1 Word (1x32bits) 5 = 4 Words (4x32bits)else = 8 Words (8x32bits) default 4 = DMA_BURST_1WORDS 5 = DMA_BURST_4WORDS 6 = DMA_BURST_8WORDS
19	0x0	DBL_BUF: 2X Double Buffering Mode (For Run-Multiple Mode with No Wrap Operations) 1 = Reload Base Address for 2X blocks (reload every other time) 0 = Reload Base Address for 1X blocks (def) (reload each time) 0 = RELOAD_FOR_1X_BLOCKS 1 = RELOAD_FOR_2X_BLOCKS

Bit	Reset	Description
18:16	0x0	WRAP: AHB Address Wrap: AHB Address wrap-around window 0=No Wrap (default) 5=Wrap on 512 word window 1=Wrap on 32 word window 6=Wrap on 1024 word window 2=Wrap on 64 word window 7=Wrap on 2048 word window 3=Wrap on 128 word window 4=Wrap on 256 word window 0 = NO_WRAP 1 = WRAP_0N_32WORDS 2 = WRAP_ON_64WORDS 3 = WRAP_ON_128WORDS 4 = WRAP_ON_256WORDS 5 = WRAP_ON_512WORDS 6 = WRAP_ON_1024WORDS 7 = WRAP_ON_2048WORDS

11.2.4.41 APBDMACHAN_CHANNEL_6_APB_PTR_0

APB-DMA-6 APB Starting Address Pointer Register

Offset: 0d8h | Read/Write: R/W | Reset: 0b00000000000000

Bit	Reset	Description
15:2	0x0	APB_BASE: APB-DMA Starting address for APB Bus: APB Base address: Upper 16 bits are fixed at 0x7000:XXXX

11.2.4.42 APBDMACHAN_CHANNEL_6_APB_SEQ_0

APB-DMA-6 APB Address Sequencer Assignments

Offset: 0dch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 3 = 64 bit Bus (RSVD) 4 = 128 bit BUS (RSVD) 0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	APB_DATA_SWAP: When enabled the data going to APB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24]} 0 = DISBALE 1 = ENABLE

Bit	Reset	Description
18:16	0x1	APB_ADDR_WRAP: APB Address Wrap-around Window 0 = No Wrap 1 = Wrap on 1 Word Window (def) 2 = Wrap on 2 Word Window 3 = Wrap on 4 Word Window 4 = Wrap on 8 Word Window 5 = Wrap on 16 Word Window 6 = Wrap on 32 Word Window 7 = Wrap on 64 Word Window (RSVD) 0 = NO_WRAP 1 = WRAP_0N_1WORDS 2 = WRAP_ON_2WORDS 3 = WRAP_ON_4WORDS 4 = WRAP_ON_8WORDS 5 = WRAP_ON_16WORDS 6 = WRAP_ON_32WORDS 7 = WRAP_ON_64WORDS

11.2.4.43 APBDMACHAN_CHANNEL_7_CSR_0

APB-DMA-7 Control

Offset: 0e0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: Enable DMA channel transfer 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: Interrupt when DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
29	0x0	HOLD: Hold this Processor until DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
28	0x0	DIR: DMA Transfer Direction 1 = AHB read to APB write 0 = APB read to AHB write 0 = AHB_WRITE 1 = AHB_READ
27	0x0	ONCE: Run Once or Run Multiple Mode (Allow Retriggering of this Channel) 1 = Run for One Block Transfer 0 = Run for Multiple Block Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK

Bit	Reset	Description
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 2 = SMP25 3 = SMP26 4 = SMP27 5 = XRQ_A 6 = XRQ_B 7 = TMR1 8 = TMR2 9 = APB_0 10 = APB_1 11 = APB_2 12 = APB_3 13 = APB_4 14 = APB_5 15 = APB_6 16 = APB_7 17 = APB_8 18 = APB_9 19 = APB_10 20 = APB_11 21 = APB_12 22 = APB_13 23 = APB_14 24 = APB_15
21	0x0	FLOW: Flow Control Enable (Synchronize Burst Transfers) 1 = Link to DRQ source 0 = Independent of DRQ request 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = I2S_2 2 = I2S_1 3 = SPD_I 4 = UI_I 5 = MIPI 6 = I2S2_2 7 = I2S2_1 8 = UART1 9 = UART2 10 = UART3 11 = SPI 12 = AC97 13 = AC Modem 14 = RSVD 15 = SPI1 16 = SPI2 17 = SPI3 18 = SPI4 19 = UART4 20 = UART5 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = NA26 27 = NA27 28 = NA28 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

11.2.4.44 APBDMACHAN_CHANNEL_7_STA_0

APB-DMA-7 Status Register

Offset: 0e4h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicate whether DMA Channel Status Active or not 0 = WAIT 1 = ACTIVE
30	RW	0x0	ISE_EOC: Write '1' to clear the flag 0 = NO_INTR 1 = INTR
29	RO	X	HALT: Holding Status of Processor 0 = NO_HALT 1 = HALT
28	RO	X	PING_PONG_STS: 0 = PING_INTR_STS 1 = PONG_INTR_STS

Bit	R/W	Reset	Description
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

11.2.4.45 APBDMACHAN_CHANNEL_7_AHB_PTR_0

APB-DMA-7 AHB Starting Address Pointer Register

Offset: 0f0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

11.2.4.46 APBDMACHAN_CHANNEL_7_AHB_SEQ_0

APB-DMA-7 AHB Address Sequencer Register

Offset: 0f4h | Read/Write: R/W | Reset: 0b00100000xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 1 = CPU 0 = COP
30:28	0x2	AHB_BUS_WIDTH: AHB Bus Width 0 = 8 bit Bus (RSVD) 1 = 16 bit Bus (RSVD) 2 = 32 bit Bus (Def) 3 = 64 bit Bus (RSVD) 4 = 128 bit Bus (RSVD) 0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	AHB_DATA_SWAP: When enabled the data going to AHB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24]} 0 = DISABLE 1 = ENABLE
26:24	0x0	AHB_BURST: AHB Burst Size DMA Burst Length (encoded) 4 = 1 Word (1x32bits) 5 = 4 Words (4x32bits)else = 8 Words (8x32bits) default 4 = DMA_BURST_1WORDS 5 = DMA_BURST_4WORDS 6 = DMA_BURST_8WORDS
19	0x0	DBL_BUF: 2X Double Buffering Mode (For Run-Multiple Mode with No Wrap Operations) 1 = Reload Base Address for 2X blocks (reload every other time) 0 = Reload Base Address for 1X blocks (def) (reload each time) 0 = RELOAD_FOR_1X_BLOCKS 1 = RELOAD_FOR_2X_BLOCKS

Bit	Reset	Description
18:16	0x0	WRAP: AHB Address Wrap: AHB Address wrap-around window 0=No Wrap (default) 5=Wrap on 512 word window 1=Wrap on 32 word window 6=Wrap on 1024 word window 2=Wrap on 64 word window 7=Wrap on 2048 word window 3=Wrap on 128 word window 4=Wrap on 256 word window 0 = NO_WRAP 1 = WRAP_0N_32WORDS 2 = WRAP_ON_64WORDS 3 = WRAP_ON_128WORDS 4 = WRAP_ON_256WORDS 5 = WRAP_ON_512WORDS 6 = WRAP_ON_1024WORDS 7 = WRAP_ON_2048WORDS

11.2.4.47 APBDMACHAN_CHANNEL_7_APB_PTR_0

APB-DMA-7 APB Starting Address Pointer Register

Offset: 0f8h | Read/Write: R/W | Reset: 0b00000000000000

Bit	Reset	Description
15:2	0x0	APB_BASE: APB-DMA Starting address for APB Bus: APB Base address: Upper 16 bits are fixed at 0x7000:XXXX

11.2.4.48 APBDMACHAN_CHANNEL_7_APB_SEQ_0

APB-DMA-7 APB Address Sequencer Assignments

Offset: 0fch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 3 = 64 bit Bus (RSVD) 4 = 128 bit BUS (RSVD) 0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	APB_DATA_SWAP: When enabled the data going to APB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24]}. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
18:16	0x1	APB_ADDR_WRAP: APB Address Wrap-around Window 0 = No Wrap 1 = Wrap on 1 Word Window (def) 2 = Wrap on 2 Word Window 3 = Wrap on 4 Word Window 4 = Wrap on 8 Word Window 5 = Wrap on 16 Word Window 6 = Wrap on 32 Word Window 7 = Wrap on 64 Word Window (RSVD) 0 = NO_WRAP 1 = WRAP_0N_1WORDS 2 = WRAP_ON_2WORDS 3 = WRAP_ON_4WORDS 4 = WRAP_ON_8WORDS 5 = WRAP_ON_16WORDS 6 = WRAP_ON_32WORDS 7 = WRAP_ON_64WORDS

11.2.4.49 APBDMACHAN_CHANNEL_8_CSR_0

APB-DMA-8 Control

Offset: 100h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: Enable DMA channel transfer 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: Interrupt when DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
29	0x0	HOLD: Hold this Processor until DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
28	0x0	DIR: DMA Transfer Direction 1 = AHB read to APB write 0 = APB read to AHB write 0 = AHB_WRITE 1 = AHB_READ
27	0x0	ONCE: Run Once or Run Multiple Mode (Allow Retriggering of this Channel) 1 = Run for One Block Transfer 0 = Run for Multiple Block Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK

Bit	Reset	Description
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 2 = SMP25 3 = SMP26 4 = SMP27 5 = XRQ_A 6 = XRQ_B 7 = TMR1 8 = TMR2 9 = APB_0 10 = APB_1 11 = APB_2 12 = APB_3 13 = APB_4 14 = APB_5 15 = APB_6 16 = APB_7 17 = APB_8 18 = APB_9 19 = APB_10 20 = APB_11 21 = APB_12 22 = APB_13 23 = APB_14 24 = APB_15
21	0x0	FLOW: Flow Control Enable (Synchronize Burst Transfers) 1 = Link to DRQ source 0 = Independent of DRQ request 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = I2S_2 2 = I2S_1 3 = SPD_I 4 = UI_I 5 = MIPI 6 = I2S2_2 7 = I2S2_1 8 = UART1 9 = UART2 10 = UART3 11 = SPI 12 = AC97 13 = AC Modem 14 = RSVD 15 = SPI1 16 = SPI2 17 = SPI3 18 = SPI4 19 = UART4 20 = UART5 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = NA26 27 = NA27 28 = NA28 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

11.2.4.50 APBDMACHAN_CHANNEL_8_STA_0

APB-DMA-8 Status Register

Offset: 104h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicate DMA Channel Status activate or not 0 = WAIT 1 = ACTIVE
30	RW	0x0	ISE_EOC: Write '1' to clear the flag 0 = NO_INTR 1 = INTR
29	RO	X	HALT: Holding Status of Processor 0 = NO_HALT 1 = HALT

Bit	R/W	Reset	Description
28	RO	X	PING_PONG_STS: 0 = PING_INTR_STS 1 = PONG_INTR_STS
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

11.2.4.51 APBDMACHAN_CHANNEL_8_AHB_PTR_0

APB-DMA-8 AHB Starting Address Pointer Register

Offset: 110h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

11.2.4.52 APBDMACHAN_CHANNEL_8_AHB_SEQ_0

APB-DMA-8 AHB Address Sequencer Register

Offset: 114h | Read/Write: R/W | Reset: 0b00100000xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 1 = CPU 0 = COP
30:28	0x2	AHB_BUS_WIDTH: AHB Bus Width 0 = 8 bit Bus (RSVD) 1 = 16 bit Bus (RSVD) 2 = 32 bit Bus (Def) 3 = 64 bit Bus (RSVD) 4 = 128 bit Bus (RSVD) 0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	AHB_DATA_SWAP: When enabled the data going to AHB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24]}. 0 = DISABLE 1 = ENABLE
26:24	0x0	AHB_BURST: AHB Burst Size DMA Burst Length (encoded) 4 = 1 Word (1x32bits) 5 = 4 Words (4x32bits)else = 8 Words (8x32bits) default 4 = DMA_BURST_1WORDS 5 = DMA_BURST_4WORDS 6 = DMA_BURST_8WORDS
19	0x0	DBL_BUF: 2X Double Buffering Mode (For Run-Multiple Mode with No Wrap Operations) 1 = Reload Base Address for 2X blocks (reload every other time) 0 = Reload Base Address for 1X blocks (def) (reload each time) 0 = RELOAD_FOR_1X_BLOCKS 1 = RELOAD_FOR_2X_BLOCKS

Bit	Reset	Description
18:16	0x0	WRAP: AHB Address Wrap: AHB Address wrap-around window 0=No Wrap (default) 5=Wrap on 512 word window 1=Wrap on 32 word window 6=Wrap on 1024 word window 2=Wrap on 64 word window 7=Wrap on 2048 word window 3=Wrap on 128 word window 4=Wrap on 256 word window 0 = NO_WRAP 1 = WRAP_0N_32WORDS 2 = WRAP_ON_64WORDS 3 = WRAP_ON_128WORDS 4 = WRAP_ON_256WORDS 5 = WRAP_ON_512WORDS 6 = WRAP_ON_1024WORDS 7 = WRAP_ON_2048WORDS

11.2.4.53 APBDMACHAN_CHANNEL_8_APB_PTR_0

APB-DMA-8 APB Starting Address Pointer Register

Offset: 118h | Read/Write: R/W | Reset: 0b00000000000000

Bit	Reset	Description
15:2	0x0	APB_BASE: APB-DMA Starting address for APB Bus: APB Base address: Upper 16 bits are fixed at 0x7000:XXXX

11.2.4.54 APBDMACHAN_CHANNEL_8_APB_SEQ_0

APB-DMA-8 APB Address Sequencer Assignments

Offset: 11ch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 3 = 64 bit Bus (RSVD) 4 = 128 bit BUS (RSVD) 0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	APB_DATA_SWAP: When enabled the data going to APB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24] }. 0 = DISBALE 1 = ENABLE

Bit	Reset	Description
18:16	0x1	APB_ADDR_WRAP: APB Address Wrap-around Window 0 = No Wrap 1 = Wrap on 1 Word Window (def) 2 = Wrap on 2 Word Window 3 = Wrap on 4 Word Window 4 = Wrap on 8 Word Window 5 = Wrap on 16 Word Window 6 = Wrap on 32 Word Window 7 = Wrap on 64 Word Window (RSVD) 0 = NO_WRAP 1 = WRAP_0N_1WORDS 2 = WRAP_ON_2WORDS 3 = WRAP_ON_4WORDS 4 = WRAP_ON_8WORDS 5 = WRAP_ON_16WORDS 6 = WRAP_ON_32WORDS 7 = WRAP_ON_64WORDS

11.2.4.55 APBDMACHAN_CHANNEL_9_CSR_0

APB-DMA-9 Control

Offset: 120h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: Enable DMA channel transfer 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: Interrupt when DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
29	0x0	HOLD: Hold this Processor until DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
28	0x0	DIR: DMA Transfer Direction 1 = AHB read to APB write 0 = APB read to AHB write 0 = AHB_WRITE 1 = AHB_READ
27	0x0	ONCE: Run Once or Run Multiple Mode (Allow Retriggering of this Channel) 1 = Run for One Block Transfer 0 = Run for Multiple Block Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK

Bit	Reset	Description
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 2 = SMP25 3 = SMP26 4 = SMP27 5 = XRQ_A 6 = XRQ_B 7 = TMR1 8 = TMR2 9 = APB_0 10 = APB_1 11 = APB_2 12 = APB_3 13 = APB_4 14 = APB_5 15 = APB_6 16 = APB_7 17 = APB_8 18 = APB_9 19 = APB_10 20 = APB_11 21 = APB_12 22 = APB_13 23 = APB_14 24 = APB_15
21	0x0	FLOW: Flow Control Enable (Synchronize Burst Transfers) 1 = Link to DRQ source 0 = Independent of DRQ request 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = I2S_2 2 = I2S_1 3 = SPD_I 4 = UI_I 5 = MIPI 6 = I2S2_2 7 = I2S2_1 8 = UART1 9 = UART2 10 = UART3 11 = SPI 12 = AC97 13 = AC Modem 14 = RSVD 15 = SPI1 16 = SPI2 17 = SPI3 18 = SPI4 19 = UART4 20 = UART5 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = NA26 27 = NA27 28 = NA28 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

11.2.4.56 APBDMACHAN_CHANNEL_9_STA_0

APB-DMA-9 Status Register

Offset: 124h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicate DMA Channel Status activate or not 0 = WAIT 1 = ACTIVE
30	RW	0x0	ISE_EOC: Write '1' to clear the flag 0 = NO_INTR 1 = INTR
29	RO	X	HALT: Holding Status of Processor 0 = NO_HALT 1 = HALT

Bit	R/W	Reset	Description
28	RO	X	PING_PONG_STS: 0 = PING_INTR_STS 1 = PONG_INTR_STS
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

11.2.4.57 APBDMACHAN_CHANNEL_9_AHB_PTR_0

APB-DMA-9 AHB Starting Address Pointer Register

Offset: 130h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

11.2.4.58 APBDMACHAN_CHANNEL_9_AHB_SEQ_0

APB-DMA-9 AHB Address Sequencer Register

Offset: 134h | Read/Write: R/W | Reset: 0b00100000xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 1 = CPU 0 = COP
30:28	0x2	AHB_BUS_WIDTH: AHB Bus Width 0 = 8 bit Bus (RSVD) 1 = 16 bit Bus (RSVD) 2 = 32 bit Bus (Def) 3 = 64 bit Bus (RSVD)4 = 128 bit Bus (RSVD) 0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	AHB_DATA_SWAP: When enabled the data going to AHB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24]} 0 = DISABLE 1 = ENABLE
26:24	0x0	AHB_BURST: AHB Burst Size DMA Burst Length (encoded) 4 = 1 Word (1x32bits) 5 = 4 Words (4x32bits)else = 8 Words (8x32bits) default 4 = DMA_BURST_1WORDS 5 = DMA_BURST_4WORDS 6 = DMA_BURST_8WORDS
19	0x0	DBL_BUF: 2X Double Buffering Mode (For Run-Multiple Mode with No Wrap Operations) 1 = Reload Base Address for 2X blocks (reload every other time) 0 = Reload Base Address for 1X blocks (def) (reload each time) 0 = RELOAD_FOR_1X_BLOCKS 1 = RELOAD_FOR_2X_BLOCKS

Bit	Reset	Description
18:16	0x0	WRAP: AHB Address Wrap: AHB Address wrap-around window 0=No Wrap (default) 5=Wrap on 512 word window 1=Wrap on 32 word window 6=Wrap on 1024 word window 2=Wrap on 64 word window 7=Wrap on 2048 word window 3=Wrap on 128 word window 4=Wrap on 256 word window 0 = NO_WRAP 1 = WRAP_0N_32WORDS 2 = WRAP_ON_64WORDS 3 = WRAP_ON_128WORDS 4 = WRAP_ON_256WORDS 5 = WRAP_ON_512WORDS 6 = WRAP_ON_1024WORDS 7 = WRAP_ON_2048WORDS

11.2.4.59 APBDMACHAN_CHANNEL_9_APB_PTR_0

APB-DMA-9 APB Starting Address Pointer Register

Offset: 138h | Read/Write: R/W | Reset: 0b00000000000000

Bit	Reset	Description
15:2	0x0	APB_BASE: APB-DMA Starting address for APB Bus: APB Base address: Upper 16 bits are fixed at 0x7000:XXXX

11.2.4.60 APBDMACHAN_CHANNEL_9_APB_SEQ_0

APB-DMA-9 APB Address Sequencer Assignments

Offset: 13ch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 3 = 64 bit Bus (RSVD) 4 = 128 bit BUS (RSVD) 0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	APB_DATA_SWAP: When enabled the data going to APB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24] } 0 = DISBALE 1 = ENABLE

Bit	Reset	Description
18:16	0x1	APB_ADDR_WRAP: APB Address Wrap-around Window 0 = No Wrap 1 = Wrap on 1 Word Window (def) 2 = Wrap on 2 Word Window 3 = Wrap on 4 Word Window 4 = Wrap on 8 Word Window 5 = Wrap on 16 Word Window 6 = Wrap on 32 Word Window 7 = Wrap on 64 Word Window (RSVD) 0 = NO_WRAP 1 = WRAP_0N_1WORDS 2 = WRAP_ON_2WORDS 3 = WRAP_ON_4WORDS 4 = WRAP_ON_8WORDS 5 = WRAP_ON_16WORDS 6 = WRAP_ON_32WORDS 7 = WRAP_ON_64WORDS

11.2.4.61 APBDMACHAN_CHANNEL_10_CSR_0

APB-DMA-10 Control

Offset: 140h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: Enable DMA channel transfer 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: Interrupt when DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
29	0x0	HOLD: Hold this Processor until DMA Block Transfer Completes 1 = Enable Hold (wait) 0 = Disable 0 = DISABLE 1 = ENABLE
28	0x0	DIR: DMA Transfer Direction 1 = AHB read to APB write 0 = APB read to AHB write 0 = AHB_WRITE 1 = AHB_READ
27	0x0	ONCE: Run Once or Run Multiple Mode (Allow Retriggering of this Channel) 1 = Run for One Block Transfer 0 = Run for Multiple Block Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK

Bit	Reset	Description
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 2 = SMP25 3 = SMP26 4 = SMP27 5 = XRQ_A 6 = XRQ_B 7 = TMR1 8 = TMR2 9 = APB_0 10 = APB_1 11 = APB_2 12 = APB_3 13 = APB_4 14 = APB_5 15 = APB_6 16 = APB_7 17 = APB_8 18 = APB_9 19 = APB_10 20 = APB_11 21 = APB_12 22 = APB_13 23 = APB_14 24 = APB_15
21	0x0	FLOW: Flow Control Enable (Synchronize Burst Transfers) 1 = Link to DRQ source 0 = Independent of DRQ request 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20:16	0x0	REQ_SEL: 0= CNTR_REQ 1 = I2S_2 2 = I2S_1 3 = SPD_I 4 = UI_I 5 = MIPI 6 = I2S2_2 7 = I2S2_1 8 = UART1 9 = UART2 10 = UART3 11 = SPI 12 = AC97 13 = AC Modem 14 = RSVD 15 = SPI1 16 = SPI2 17 = SPI3 18 = SPI4 19 = UART4 20 = UART5 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = NA26 27 = NA27 28 = NA28 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

11.2.4.62 APBDMACHAN_CHANNEL_10_STA_0

APB-DMA-10 Status Register

Offset: 144h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicate DMA Channel Status activate or not 0 = WAIT 1 = ACTIVE
30	RW	0x0	ISE_EOC: Write '1' to clear the flag 0 = NO_INTR 1 = INTR
29	RO	X	HALT: Holding Status of Processor 0 = NO_HALT 1 = HALT

Bit	R/W	Reset	Description
28	RO	X	PING_PONG_STS: 0 = PING_INTR_STS 1 = PONG_INTR_STS
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

11.2.4.63 APBDMACHAN_CHANNEL_10_AHB_PTR_0

APB-DMA-10 AHB Starting Address Pointer Register

Offset: 150h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

11.2.4.64 APBDMACHAN_CHANNEL_10_AHB_SEQ_0

APB-DMA-10 AHB Address Sequencer Register

Offset: 154h | Read/Write: R/W | Reset: 0b00100000xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 1 = CPU 0 = COP
30:28	0x2	AHB_BUS_WIDTH: AHB Bus Width 0 = 8 bit Bus (RSVD) 1 = 16 bit Bus (RSVD) 2 = 32 bit Bus (Def) 3 = 64 bit Bus (RSVD)4 = 128 bit Bus (RSVD) 0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	AHB_DATA_SWAP: When enabled the data going to AHB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24]} 0 = DISABLE 1 = ENABLE
26:24	0x0	AHB_BURST: AHB Burst Size DMA Burst Length (encoded) 4 = 1 Word (1x32bits) 5 = 4 Words (4x32bits)else = 8 Words (8x32bits) default 4 = DMA_BURST_1WORDS 5 = DMA_BURST_4WORDS 6 = DMA_BURST_8WORDS
19	0x0	DBL_BUF: 2X Double Buffering Mode (For Run-Multiple Mode with No Wrap Operations) 1 = Reload Base Address for 2X blocks (reload every other time) 0 = Reload Base Address for 1X blocks (def) (reload each time) 0 = RELOAD_FOR_1X_BLOCKS 1 = RELOAD_FOR_2X_BLOCKS

Bit	Reset	Description
18:16	0x0	WRAP: AHB Address Wrap: AHB Address wrap-around window 0=No Wrap (default) 5=Wrap on 512 word window 1=Wrap on 32 word window 6=Wrap on 1024 word window 2=Wrap on 64 word window 7=Wrap on 2048 word window 3=Wrap on 128 word window 4=Wrap on 256 word window 0 = NO_WRAP 1 = WRAP_0N_32WORDS 2 = WRAP_ON_64WORDS 3 = WRAP_ON_128WORDS 4 = WRAP_ON_256WORDS 5 = WRAP_ON_512WORDS 6 = WRAP_ON_1024WORDS 7 = WRAP_ON_2048WORDS

11.2.4.65 APBDMACHAN_CHANNEL_10_APB_PTR_0

APB-DMA-10 APB Starting Address Pointer Register

Offset: 158h | Read/Write: R/W | Reset: 0b00000000000000

Bit	Reset	Description
15:2	0x0	APB_BASE: APB-DMA Starting address for APB Bus: APB Base address: Upper 16 bits are fixed at 0x7000:XXXX

11.2.4.66 APBDMACHAN_CHANNEL_10_APB_SEQ_0

APB-DMA-10 APB Address Sequencer Assignments

Offset: 15ch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 3 = 64 bit Bus (RSVD) 4 = 128 bit BUS (RSVD) 0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	APB_DATA_SWAP: When enabled the data going to APB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24]} 0 = DISBALE 1 = ENABLE

Bit	Reset	Description
18:16	0x1	APB_ADDR_WRAP: APB Address Wrap-around Window 0 = No Wrap 1 = Wrap on 1 Word Window (def) 2 = Wrap on 2 Word Window 3 = Wrap on 4 Word Window 4 = Wrap on 8 Word Window 5 = Wrap on 16 Word Window 6 = Wrap on 32 Word Window 7 = Wrap on 64 Word Window (RSVD) 0 = NO_WRAP 1 = WRAP_0N_1WORDS 2 = WRAP_ON_2WORDS 3 = WRAP_ON_4WORDS 4 = WRAP_ON_8WORDS 5 = WRAP_ON_16WORDS 6 = WRAP_ON_32WORDS 7 = WRAP_ON_64WORDS

11.2.4.67 APBDMACHAN_CHANNEL_11_CSR_0

APB-DMA-11 Control

Offset: 160h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: Enable DMA channel transfer 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: Interrupt when DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
29	0x0	HOLD: Hold this Processor until DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
28	0x0	DIR: DMA Transfer Direction 1 = AHB read to APB write 0 = APB read to AHB write 0 = AHB_WRITE 1 = AHB_READ
27	0x0	ONCE: Run Once or Run Multiple Mode (Allow Retriggering of this Channel) 1 = Run for One Block Transfer 0 = Run for Multiple Block Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK

Bit	Reset	Description
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 2 = SMP25 3 = SMP26 4 = SMP27 5 = XRQ_A 6 = XRQ_B 7 = TMR1 8 = TMR2 9 = APB_0 10 = APB_1 11 = APB_2 12 = APB_3 13 = APB_4 14 = APB_5 15 = APB_6 16 = APB_7 17 = APB_8 18 = APB_9 19 = APB_10 20 = APB_11 21 = APB_12 22 = APB_13 23 = APB_14 24 = APB_15
21	0x0	FLOW: Flow Control Enable (Synchronize Burst Transfers) 1 = Link to DRQ source 0 = Independent of DRQ request 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = I2S_2 2 = I2S_1 3 = SPD_I 4 = UI_I 5 = MIPI 6 = I2S2_2 7 = I2S2_1 8 = UART1 9 = UART2 10 = UART3 11 = SPI 12 = AC97 13 = AC Modem 14 = RSVD 15 = SPI1 16 = SPI2 17 = SPI3 18 = SPI4 19 = UART4 20 = UART5 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = NA26 27 = NA27 28 = NA28 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

11.2.4.68 APBDMACHAN_CHANNEL_11_STA_0

APB-DMA-11 Status Register

Offset: 164h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicate DMA Channel Status activate or waiting 0 = WAIT 1 = ACTIVE
30	RW	0x0	ISE_EOC: Write '1' to clear the flag 0 = NO_INTR 1 = INTR
29	RO	X	HALT: Holding Status of Processor 0 = NO_HALT 1 = HALT

Bit	R/W	Reset	Description
28	RO	X	PING_PONG_STS: 0 = PING_INTR_STS 1 = PONG_INTR_STS
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

11.2.4.69 APBDMACHAN_CHANNEL_11_AHB_PTR_0

APB-DMA-11 AHB Starting Address Pointer Register

Offset: 170h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

11.2.4.70 APBDMACHAN_CHANNEL_11_AHB_SEQ_0

APB-DMA-11 AHB Address Sequencer Register

Offset: 174h | Read/Write: R/W | Reset: 0b00100000xxxxx000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 1 = CPU 0 = COP
30:28	0x2	AHB_BUS_WIDTH: AHB Bus Width 0 = 8 bit Bus (RSVD) 1 = 16 bit Bus (RSVD) 2 = 32 bit Bus (Def) 3 = 64 bit Bus (RSVD)4 = 128 bit Bus (RSVD) 0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	AHB_DATA_SWAP: When enabled the data going to AHB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24]} 0 = DISABLE 1 = ENABLE
26:24	0x0	AHB_BURST: AHB Burst Size DMA Burst Length (encoded) 4 = 1 Word (1x32bits) 5 = 4 Words (4x32bits) else = 8 Words (8x32bits) default 4 = DMA_BURST_1WORDS 5 = DMA_BURST_4WORDS 6 = DMA_BURST_8WORDS

Bit	Reset	Description
18:16	0x0	WRAP: AHB Address Wrap: AHB Address wrap-around window 0=No Wrap (default) 5=Wrap on 512 word window 1=Wrap on 32 word window 6=Wrap on 1024 word window 2=Wrap on 64 word window 7=Wrap on 2048 word window 3=Wrap on 128 word window 4=Wrap on 256 word window 0 = NO_WRAP 1 = WRAP_0N_32WORDS 2 = WRAP_ON_64WORDS 3 = WRAP_ON_128WORDS 4 = WRAP_ON_256WORDS 5 = WRAP_ON_512WORDS 6 = WRAP_ON_1024WORDS 7 = WRAP_ON_2048WORDS

11.2.4.71 APBDMACHAN_CHANNEL_11_APB_PTR_0

APB-DMA-11 APB Starting Address Pointer Register

Offset: 178h | Read/Write: R/W | Reset: 0b00000000000000

Bit	Reset	Description
15:2	0x0	APB_BASE: APB-DMA Starting address for APB Bus: APB Base address: Upper 16 bits are fixed at 0x7000:XXXX

11.2.4.72 APBDMACHAN_CHANNEL_11_APB_SEQ_0

APB-DMA-11 APB Address Sequencer Assignments

Offset: 17ch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 3 = 64 bit Bus (RSVD) 4 = 128 bit BUS (RSVD) 0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	APB_DATA_SWAP: When enabled the data going to APB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24]} 0 = DISBALE 1 = ENABLE

Bit	Reset	Description
18:16	0x1	APB_ADDR_WRAP: APB Address Wrap-around Window 0 = No Wrap 1 = Wrap on 1 Word Window (def) 2 = Wrap on 2 Word Window 3 = Wrap on 4 Word Window 4 = Wrap on 8 Word Window 5 = Wrap on 16 Word Window 6 = Wrap on 32 Word Window 7 = Wrap on 64 Word Window (RSVD) 0 = NO_WRAP 1 = WRAP_0N_1WORDS 2 = WRAP_ON_2WORDS 3 = WRAP_ON_4WORDS 4 = WRAP_ON_8WORDS 5 = WRAP_ON_16WORDS 6 = WRAP_ON_32WORDS 7 = WRAP_ON_64WORDS

11.2.4.73 APBDMACHAN_CHANNEL_12_CSR_0

APB-DMA-12 Control

Offset: 180h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: Enable DMA channel transfer 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: Interrupt when DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
29	0x0	HOLD: Hold this Processor until DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
28	0x0	DIR: DMA Transfer Direction 1 = AHB read to APB write 0 = APB read to AHB write 0 = AHB_WRITE 1 = AHB_READ
27	0x0	ONCE: Run Once or Run Multiple Mode (Allow Retriggering of this Channel) 1 = Run for One Block Transfer 0 = Run for Multiple Block Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK

Bit	Reset	Description
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 2 = SMP25 3 = SMP26 4 = SMP27 5 = XRQ_A 6 = XRQ_B 7 = TMR1 8 = TMR2 9 = APB_0 10 = APB_1 11 = APB_2 12 = APB_3 13 = APB_4 14 = APB_5 15 = APB_6 16 = APB_7 17 = APB_8 18 = APB_9 19 = APB_10 20 = APB_11 21 = APB_12 22 = APB_13 23 = APB_14 24 = APB_15
21	0x0	FLOW: Flow Control Enable (Synchronize Burst Transfers) 1 = Link to DRQ source 0 = Independent of DRQ request 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = I2S_2 2 = I2S_1 3 = SPD_I 4 = UI_I 5 = MIPI 6 = I2S2_2 7 = I2S2_1 8 = UART1 9 = UART2 10 = UART3 11 = SPI 12 = AC97 13 = AC Modem 14 = RSVD 15 = SPI1 16 = SPI2 17 = SPI3 18 = SPI4 19 = UART4 20 = UART5 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = NA26 27 = NA27 28 = NA28 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

11.2.4.74 APBDMACHAN_CHANNEL_12_STA_0

APB-DMA-12 Status Register

Offset: 184h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicate DMA Channel Status activate or not 0 = WAIT 1 = ACTIVE
30	RW	0x0	ISE_EOC: Write '1' to clear the flag 0 = NO_INTR 1 = INTR
29	RO	X	HALT: Holding Status of Processor 0 = NO_HALT 1 = HALT

Bit	R/W	Reset	Description
28	RO	X	PING_PONG_STS: 0 = PING_INTR_STS 1 = PONG_INTR_STS
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

11.2.4.75 APBDMACHAN_CHANNEL_12_AHB_PTR_0

APB-DMA-12 AHB Starting Address Pointer Register

Offset: 190h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

11.2.4.76 APBDMACHAN_CHANNEL_12_AHB_SEQ_0

APB-DMA-12 AHB Address Sequencer Register

Offset: 194h | Read/Write: R/W | Reset: 0b00100000xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 1 = CPU 0 = COP
30:28	0x2	AHB_BUS_WIDTH: AHB Bus Width 0 = 8 bit Bus (RSVD) 1 = 16 bit Bus (RSVD) 2 = 32 bit Bus (Def) 3 = 64 bit Bus (RSVD)4 = 128 bit Bus (RSVD) 0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	AHB_DATA_SWAP: When enabled the data going to AHB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24] } 0 = DISABLE 1 = ENABLE
26:24	0x0	AHB_BURST: AHB Burst Size DMA Burst Length (encoded) 4 = 1 Word (1x32bits) 5 = 4 Words (4x32bits)else = 8 Words (8x32bits) default 4 = DMA_BURST_1WORDS 5 = DMA_BURST_4WORDS 6 = DMA_BURST_8WORDS
19	0x0	DBL_BUF: 2X Double Buffering Mode (For Run-Multiple Mode with No Wrap Operations) 1 = Reload Base Address for 2X blocks (reload every other time) 0 = Reload Base Address for 1X blocks (def) (reload each time) 0 = RELOAD_FOR_1X_BLOCKS 1 = RELOAD_FOR_2X_BLOCKS

Bit	Reset	Description
18:16	0x0	WRAP: AHB Address Wrap: AHB Address wrap-around window 0=No Wrap (default) 5=Wrap on 512 word window 1=Wrap on 32 word window 6=Wrap on 1024 word window 2=Wrap on 64 word window 7=Wrap on 2048 word window 3=Wrap on 128 word window 4=Wrap on 256 word window 0 = NO_WRAP 1 = WRAP_0N_32WORDS 2 = WRAP_ON_64WORDS 3 = WRAP_ON_128WORDS 4 = WRAP_ON_256WORDS 5 = WRAP_ON_512WORDS 6 = WRAP_ON_1024WORDS 7 = WRAP_ON_2048WORDS

11.2.4.77 APBDMACHAN_CHANNEL_12_APB_PTR_0

APB-DMA-12 APB Starting Address Pointer Register

Offset: 198h | Read/Write: R/W | Reset: 0b00000000000000

Bit	Reset	Description
15:2	0x0	APB_BASE: APB-DMA Starting address for APB Bus: APB Base address: Upper 16 bits are fixed at 0x7000:XXXX

11.2.4.78 APBDMACHAN_CHANNEL_12_APB_SEQ_0

APB-DMA-12 APB Address Sequencer Assignments

Offset: 19ch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 3 = 64 bit Bus (RSVD) 4 = 128 bit BUS (RSVD) 0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	APB_DATA_SWAP: When enabled the data going to APB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24]} 0 = DISBALE 1 = ENABLE

Bit	Reset	Description
18:16	0x1	APB_ADDR_WRAP: APB Address Wrap-around Window 0 = No Wrap 1 = Wrap on 1 Word Window (def) 2 = Wrap on 2 Word Window 3 = Wrap on 4 Word Window 4 = Wrap on 8 Word Window 5 = Wrap on 16 Word Window 6 = Wrap on 32 Word Window 7 = Wrap on 64 Word Window (RSVD) 0 = NO_WRAP 1 = WRAP_0N_1WORDS 2 = WRAP_ON_2WORDS 3 = WRAP_ON_4WORDS 4 = WRAP_ON_8WORDS 5 = WRAP_ON_16WORDS 6 = WRAP_ON_32WORDS 7 = WRAP_ON_64WORDS

11.2.4.79 APBDMACHAN_CHANNEL_13_CSR_0

APB-DMA-13 Control

Offset: 1a0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: Enable DMA channel transfer 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: Interrupt when DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
29	0x0	HOLD: Hold this Processor until DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
28	0x0	DIR: DMA Transfer Direction 1 = AHB read to APB write 0 = APB read to AHB write 0= AHB_WRITE 1 = AHB_READ
27	0x0	ONCE: Run Once or Run Multiple Mode (Allow Retriggering of this Channel) 1 = Run for One Block Transfer 0 = Run for Multiple Block Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK

Bit	Reset	Description
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 2 = SMP25 3 = SMP26 4 = SMP27 5 = XRQ_A 6 = XRQ_B 7 = TMR1 8 = TMR2 9 = APB_0 10 = APB_1 11 = APB_2 12 = APB_3 13 = APB_4 14 = APB_5 15 = APB_6 16 = APB_7 17 = APB_8 18 = APB_9 19 = APB_10 20 = APB_11 21 = APB_12 22 = APB_13 23 = APB_14 24 = APB_15
21	0x0	FLOW: Flow Control Enable (Synchronize Burst Transfers) 1 = Link to DRQ source 0 = Independent of DRQ request 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = I2S_2 2 = I2S_1 3 = SPD_I 4 = UI_I 5 = MIPI 6 = I2S2_2 7 = I2S2_1 8 = UART1 9 = UART2 10 = UART3 11 = SPI 12 = AC97 13 = AC Modem 14 = RSVD 15 = SPI1 16 = SPI2 17 = SPI3 18 = SPI4 19 = UART4 20 = UART5 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = NA26 27 = NA27 28 = NA28 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

11.2.4.80 APBDMACHAN_CHANNEL_13_STA_0

APB-DMA-13 Status Register

Offset: 1a4h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicate DMA Channel Status activate or not 0 = WAIT 1 = ACTIVE
30	RW	0x0	ISE_EOC: Write '1' to clear the flag 0 = NO_INTR 1 = INTR
29	RO	X	HALT: Holding Status of Processor 0 = NO_HALT 1 = HALT

Bit	R/W	Reset	Description
28	RO	X	PING_PONG_STS: 0 = PING_INTR_STS 1 = PONG_INTR_STS
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

11.2.4.81 APBDMACHAN_CHANNEL_13_AHB_PTR_0

APB-DMA-13 AHB Starting Address Pointer Register

Offset: 1b0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

11.2.4.82 APBDMACHAN_CHANNEL_13_AHB_SEQ_0

APB-DMA-13 AHB Address Sequencer Register

Offset: 1b4h | Read/Write: R/W | Reset: 0b00100000xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 1 = CPU 0 = COP
30:28	0x2	AHB_BUS_WIDTH: AHB Bus Width 0 = 8 bit Bus (RSVD) 1 = 16 bit Bus (RSVD) 2 = 32 bit Bus (Def) 3 = 64 bit Bus (RSVD)4 = 128 bit Bus (RSVD) 0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	AHB_DATA_SWAP: When enabled the data going to AHB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24] } 0 = DISABLE 1 = ENABLE
26:24	0x0	AHB_BURST: AHB Burst Size DMA Burst Length (encoded) 4 = 1 Word (1x32bits) 5 = 4 Words (4x32bits)else = 8 Words (8x32bits) default 4 = DMA_BURST_1WORDS 5 = DMA_BURST_4WORDS 6 = DMA_BURST_8WORDS
19	0x0	DBL_BUF: 2X Double Buffering Mode (For Run-Multiple Mode with No Wrap Operations) 1 = Reload Base Address for 2X blocks (reload every other time) 0 = Reload Base Address for 1X blocks (def) (reload each time) 0 = RELOAD_FOR_1X_BLOCKS 1 = RELOAD_FOR_2X_BLOCKS

Bit	Reset	Description
18:16	0x0	WRAP: AHB Address Wrap: AHB Address wrap-around window 0=No Wrap (default) 5=Wrap on 512 word window 1=Wrap on 32 word window 6=Wrap on 1024 word window 2=Wrap on 64 word window 7=Wrap on 2048 word window 3=Wrap on 128 word window 4=Wrap on 256 word window 0 = NO_WRAP 1 = WRAP_0N_32WORDS 2 = WRAP_ON_64WORDS 3 = WRAP_ON_128WORDS 4 = WRAP_ON_256WORDS 5 = WRAP_ON_512WORDS 6 = WRAP_ON_1024WORDS 7 = WRAP_ON_2048WORDS

11.2.4.83 APBDMACHAN_CHANNEL_13_APB_PTR_0

APB-DMA-13 APB Starting Address Pointer Register

Offset: 1b8h | Read/Write: R/W | Reset: 0b00000000000000

Bit	Reset	Description
15:2	0x0	APB_BASE: APB-DMA Starting address for APB Bus: APB Base address: Upper 16 bits are fixed at 0x7000:XXXX

11.2.4.84 APBDMACHAN_CHANNEL_13_APB_SEQ_0

APB-DMA-13 APB Address Sequencer Assignments

Offset: 1bch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 3 = 64 bit Bus (RSVD) 4 = 128 bit BUS (RSVD) 0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	APB_DATA_SWAP: When enabled the data going to APB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24]} 0= DISBALE 1 = ENABLE

Bit	Reset	Description
18:16	0x1	APB_ADDR_WRAP: APB Address Wrap-around Window 0 = No Wrap 1 = Wrap on 1 Word Window (def) 2 = Wrap on 2 Word Window 3 = Wrap on 4 Word Window 4 = Wrap on 8 Word Window 5 = Wrap on 16 Word Window 6 = Wrap on 32 Word Window 7 = Wrap on 64 Word Window (RSVD) 0 = NO_WRAP 1 = WRAP_0N_1WORDS 2 = WRAP_ON_2WORDS 3 = WRAP_ON_4WORDS 4 = WRAP_ON_8WORDS 5 = WRAP_ON_16WORDS 6 = WRAP_ON_32WORDS 7 = WRAP_ON_64WORDS

11.2.4.85 APBDMACHAN_CHANNEL_14_CSR_0

APB-DMA-14 Control

Offset: 1c0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: Enable DMA channel transfer 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: Interrupt when DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
29	0x0	HOLD: Hold this Processor until DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
28	0x0	DIR: DMA Transfer Direction 1 = AHB read to APB write 0 = APB read to AHB write 0 = AHB_WRITE 1 = AHB_READ
27	0x0	ONCE: Run Once or Run Multiple Mode (Allow Retriggering of this Channel) 1 = Run for One Block Transfer 0 = Run for Multiple Block Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK

Bit	Reset	Description
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 2 = SMP25 3 = SMP26 4 = SMP27 5 = XRQ_A 6 = XRQ_B 7 = TMR1 8 = TMR2 9 = APB_0 10 = APB_1 11 = APB_2 12 = APB_3 13 = APB_4 14 = APB_5 15 = APB_6 16 = APB_7 17 = APB_8 18 = APB_9 19 = APB_10 20 = APB_11 21 = APB_12 22 = APB_13 23 = APB_14 24 = APB_15
21	0x0	FLOW: Flow Control Enable (Synchronize Burst Transfers) 1 = Link to DRQ source 0 = Independent of DRQ request 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = I2S_2 2 = I2S_1 3 = SPD_I 4 = UI_I 5 = MIPI 6 = I2S2_2 7 = I2S2_1 8 = UART1 9 = UART2 10 = UART3 11 = SPI 12 = AC97 13 = AC Modem 14 = RSVD 15 = SPI1 16 = SPI2 17 = SPI3 18 = SPI4 19 = UART4 20 = UART5 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = NA26 27 = NA27 28 = NA28 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

11.2.4.86 APBDMACHAN_CHANNEL_14_STA_0

APB-DMA-14 Status Register

Offset: 1c4h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicate DMA Channel Status activate or not 0 = WAIT 1 = ACTIVE
30	RW	0x0	ISE_EOC: Write '1' to clear the flag 0 = NO_INTR 1 = INTR
29	RO	X	HALT: Holding Status of Processor 0 = NO_HALT 1 = HALT

Bit	R/W	Reset	Description
28	RO	X	PING_PONG_STS: 0 = PING_INTR_STS 1 = PONG_INTR_STS
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

11.2.4.87 APBDMACHAN_CHANNEL_14_AHB_PTR_0

APB-DMA-14 AHB Starting Address Pointer Register

Offset: 1d0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

11.2.4.88 APBDMACHAN_CHANNEL_14_AHB_SEQ_0

APB-DMA-14 AHB Address Sequencer Register

Offset: 1d4h | Read/Write: R/W | Reset: 0b00100000xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 1 = CPU 0 = COP
30:28	0x2	AHB_BUS_WIDTH: AHB Bus Width 0 = 8 bit Bus (RSVD) 1 = 16 bit Bus (RSVD) 2 = 32 bit Bus (Def) 3 = 64 bit Bus (RSVD)4 = 128 bit Bus (RSVD) 0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	AHB_DATA_SWAP: When enabled the data going to AHB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24]} 0 = DISABLE 1 = ENABLE
26:24	0x0	AHB_BURST: AHB Burst Size DMA Burst Length (encoded) 4 = 1 Word (1x32bits) 5 = 4 Words (4x32bits)else = 8 Words (8x32bits) default 4 = DMA_BURST_1WORDS 5 = DMA_BURST_4WORDS 6 = DMA_BURST_8WORDS
19	0x0	DBL_BUF: 2X Double Buffering Mode (For Run-Multiple Mode with No Wrap Operations) 1 = Reload Base Address for 2X blocks (reload every other time) 0 = Reload Base Address for 1X blocks (def) (reload each time) 0 = RELOAD_FOR_1X_BLOCKS 1 = RELOAD_FOR_2X_BLOCKS

Bit	Reset	Description
18:16	0x0	WRAP: AHB Address Wrap: AHB Address wrap-around window 0=No Wrap (default) 5=Wrap on 512 word window 1=Wrap on 32 word window 6=Wrap on 1024 word window 2=Wrap on 64 word window 7=Wrap on 2048 word window 3=Wrap on 128 word window 4=Wrap on 256 word window 0 = NO_WRAP 1 = WRAP_0N_32WORDS 2 = WRAP_ON_64WORDS 3 = WRAP_ON_128WORDS 4 = WRAP_ON_256WORDS 5 = WRAP_ON_512WORDS 6 = WRAP_ON_1024WORDS 7 = WRAP_ON_2048WORDS

11.2.4.89 APBDMACHAN_CHANNEL_14_APB_PTR_0

APB-DMA-14 APB Starting Address Pointer Register

Offset: 1d8h | Read/Write: R/W | Reset: 0b00000000000000

Bit	Reset	Description
15:2	0x0	APB_BASE: APB-DMA Starting address for APB Bus: APB Base address: Upper 16 bits are fixed at 0x7000:XXXX

11.2.4.90 APBDMACHAN_CHANNEL_14_APB_SEQ_0

APB-DMA-14 APB Address Sequencer Assignments

Offset: 1dch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 3 = 64 bit Bus (RSVD) 4 = 128 bit BUS (RSVD) 0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	APB_DATA_SWAP: When enabled the data going to APB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24]} 0 = DISBALE 1 = ENABLE

Bit	Reset	Description
18:16	0x1	APB_ADDR_WRAP: APB Address Wrap-around Window 0 = No Wrap 1 = Wrap on 1 Word Window (def) 2 = Wrap on 2 Word Window 3 = Wrap on 4 Word Window 4 = Wrap on 8 Word Window 5 = Wrap on 16 Word Window 6 = Wrap on 32 Word Window 7 = Wrap on 64 Word Window (RSVD) 0 = NO_WRAP 1 = WRAP_0N_1WORDS 2 = WRAP_ON_2WORDS 3 = WRAP_ON_4WORDS 4 = WRAP_ON_8WORDS 5 = WRAP_ON_16WORDS 6 = WRAP_ON_32WORDS 7 = WRAP_ON_64WORDS

11.2.4.91 APBDMACHAN_CHANNEL_15_CSR_0

APB-DMA-15 Control

Offset: 1e0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: Enable DMA channel transfer 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: Interrupt when DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
29	0x0	HOLD: Hold this Processor until DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
28	0x0	DIR: DMA Transfer Direction 1 = AHB read to APB write 0 = APB read to AHB write 0 = AHB_WRITE 1 = AHB_READ
27	0x0	ONCE: Run Once or Run Multiple Mode (Allow Retriggering of this Channel) 1 = Run for One Block Transfer 0 = Run for Multiple Block Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK

Bit	Reset	Description
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 2 = SMP25 3 = SMP26 4 = SMP27 5 = XRQ_A 6 = XRQ_B 7 = TMR1 8 = TMR2 9 = APB_0 10 = APB_1 11 = APB_2 12 = APB_3 13 = APB_4 14 = APB_5 15 = APB_6 16 = APB_7 17 = APB_8 18 = APB_9 19 = APB_10 20 = APB_11 21 = APB_12 22 = APB_13 23 = APB_14 24 = APB_15
21	0x0	FLOW: Flow Control Enable (Synchronize Burst Transfers) 1 = Link to DRQ source 0 = Independent of DRQ request 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = I2S_2 2 = I2S_1 3 = SPD_I 4 = UI_I 5 = MIPI 6 = I2S2_2 7 = I2S2_1 8 = UART1 9 = UART2 10 = UART3 11 = SPI 12 = AC97 13 = AC Modem 14 = RSVD 15 = SPI1 16 = SPI2 17 = SPI3 18 = SPI4 19 = UART4 20 = UART5 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR26 = NA26 27 = NA27 28 = NA28 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

11.2.4.92 APBDMACHAN_CHANNEL_15_STA_0

APB-DMA-15 Status Register

Offset: 1e4h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicate DMA Channel Status activate or not 0 = WAIT 1 = ACTIVE
30	RW	0x0	ISE_EOC: Write '1' to clear the flag 0 = NO_INTR 1 = INTR
29	RO	X	HALT: Holding Status of Processor 0 = NO_HALT 1 = HALT
28	RO	X	PING_PONG_STS: 0 = PING_INTR_STS 1 = PONG_INTR_STS

Bit	R/W	Reset	Description
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

11.2.4.93 APBDMACHAN_CHANNEL_15_AHB_PTR_0

APB-DMA-15 AHB Starting Address Pointer Register

Offset: 1f0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

11.2.4.94 APBDMACHAN_CHANNEL_15_AHB_SEQ_0

APB-DMA-15 AHB Address Sequencer Register

Offset: 1f4h | Read/Write: R/W | Reset: 0b00100000xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 1 = CPU 0 = COP
30:28	0x2	AHB_BUS_WIDTH: AHB Bus Width 0 = 8 bit Bus (RSVD) 1 = 16 bit Bus (RSVD) 2 = 32 bit Bus (Def) 3 = 64 bit Bus (RSVD)4 = 128 bit Bus (RSVD) 0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	AHB_DATA_SWAP: When enabled the data going to AHB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24]} 0 = DISABLE 1 = ENABLE
26:24	0x0	AHB_BURST: AHB Burst Size DMA Burst Length (encoded) 4 = 1 Word (1x32bits) 5 = 4 Words (4x32bits)else = 8 Words (8x32bits) default 4 = DMA_BURST_1WORDS 5 = DMA_BURST_4WORDS 6 = DMA_BURST_8WORDS
19	0x0	DBL_BUF: 2X Double Buffering Mode (For Run-Multiple Mode with No Wrap Operations) 1 = Reload Base Address for 2X blocks (reload every other time) 0 = Reload Base Address for 1X blocks (def) (reload each time) 0 = RELOAD_FOR_1X_BLOCKS 1 = RELOAD_FOR_2X_BLOCKS

Bit	Reset	Description
18:16	0x0	WRAP: AHB Address Wrap: AHB Address wrap-around window 0=No Wrap (default) 5=Wrap on 512 word window 1=Wrap on 32 word window 6=Wrap on 1024 word window 2=Wrap on 64 word window 7=Wrap on 2048 word window 3=Wrap on 128 word window 4=Wrap on 256 word window 0 = NO_WRAP 1 = WRAP_0N_32WORDS 2 = WRAP_ON_64WORDS 3 = WRAP_ON_128WORDS 4 = WRAP_ON_256WORDS 5 = WRAP_ON_512WORDS 6 = WRAP_ON_1024WORDS 7 = WRAP_ON_2048WORDS

11.2.4.95 APBDMACHAN_CHANNEL_15_APB_PTR_0

APB-DMA-15 APB Starting Address Pointer Register

Offset: 1f8h | Read/Write: R/W | Reset: 0b00000000000000

Bit	Reset	Description
15:2	0x0	APB_BASE: APB-DMA Starting address for APB Bus: APB Base address: Upper 16 bits are fixed at 0x7000:XXXX

11.2.4.96 APBDMACHAN_CHANNEL_15_APB_SEQ_0

APB-DMA-15 APB Address Sequencer Assignments

Offset: 1fch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 3 = 64 bit Bus (RSVD) 4 = 128 bit BUS (RSVD) 0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	APB_DATA_SWAP: When enabled the data going to APB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24]}. 0 = DISABLE 1 = ENABLE
18:16	0x1	APB_ADDR_WRAP: APB Address Wrap-around 0 = NO_WRAP 1 = WRAP_0N_1WORDS 2 = WRAP_ON_2WORDS 3 = WRAP_ON_4WORDS 4 = WRAP_ON_8WORDS 5 = WRAP_ON_16WORDS 6 = WRAP_ON_32WORDS 7 = WRAP_ON_64WORDS

12.0 CPU

The NVIDIA® Tegra® 2 Processor CPU complex contains dual ARM Cortex™-A9 CPUs and a PL310 L2 Cache controller. The L2 cache controller is described in another section under L2 Cache Controller.

The CPUs are dual ARM Cortex-A9 MPCore™ processors, which is the SMP variant of the Cortex-A9 CPU (there is also a single core variant). These include the Snoop Control Unit (SCU) which maintains coherency between the L1 cache contents of the two CPUs.

This section does not document the CPU itself; for that you will need the ARM documentation which is available from ARM:

Cortex-A9 MPCore
Revision: r1p0
Technical Reference Manual

Published by ARM Limited, document number ARM DDI 0407C.

Note:	Current version at this time. You should periodically look for updates and refer to the latest available version of this material.

Implementation Details

Many aspects of the Cortex-A9 may be configured for a specific implementation. The Tegra 2 Series implementation has the following features:

- r1p1 revision level (see note below)
- 32 Kbyte I-cache
- 32 Kbyte D-cache
- 128 external interrupt inputs (SPIs)
- VFPv3-D16 FPU, high performance scalar floating point unit
- TLB is 64+4 entries
- ACP present

Note:	Revision level reported is r1p0 in the CPU ID register, but the logic contains the fix for ARM Erratum #677567 which is the sole difference between r1p0 and r1p1. Silicon revision A04 onwards also contains the fix for ARM Erratum #745320. The Cortex-A9 Floating-Pont Unit has its own separate Technical Reference Manual, available from ARM.

12.1 CPU Timers

The Cortex-A9 processor contains built in timers. These may be used by software, but are subject to some restrictions and warnings that the system timers external to the CPU complex are not.

These are:

- The timers are always clocked at the CPU clock speed divided by four. Therefore they will have to be re-programmed to maintain real time when the CPU clock frequency is changed.
- There is no hardware interaction between these timers and the flow controller, and the external interrupt unit, unlike the system timers. They are entirely internal to the Cortex-A9 processor.
- If the CPU is powered off, then they will stop counting and their contents will be lost unless saved and restored. They cannot be used to wake the CPU from a power down.

Refer to the Cortex-A9 documentation for further details of these timers.

13.0 FLOW CONTROLLER

The flow controller provides a mechanism to halt processors conditionally or unconditionally:

- Conditional halt/resume is based on a variety of events, including timers, external trigger, and internal clocks.
- COP (AVP) is halted by extending a bus cycle.
- CPU is halted in one of two ways:
  - Software Interaction: software issues the WFE instruction to stop the processor; the flow controller asserts the EVENT input of the CPU to restart it.
  - Stop the clock [Legacy]: should never be done for application SW.
- Clocks to the processors can be halted when halted through flow control.

13.1 Flow Controller Registers

The EVENTS register is replicated three times, once each for COP (AVP), CPU0 and CPU1.

The fields and enums have the following interpretation:

- MODE defines how the flow controller logic operates; the reset value is 0x0 (does nothing).
- Enumerated values for the field MODE are defined below.

Field Mode	Enumerated Value	Description
FLOW_MODE_NONE	0	No flow control
FLOW_MODE_RUN_AND_INT	1	Keep running but generate interrupt when event conditions met
FLOW_MODE_WAITEVENT	2	Stop running until event conditions met
FLOW_MODE_WAITEVENT_AND_INT	3	Same as FLOW_MODE_WAITEVENT but generate an interrupt when resumed
FLOW_MODE_STOP_UNTIL_IRQ	4	Stop until an interrupt controller interrupt occurs
FLOW_MODE_STOP_UNTIL_IRQ_AND_INT	5	Same as FLOW_MODE_STOP_UNTIL_INT but generate another interrupt when resumed
FLOW_MODE_STOP_UNTIL_EVENT_AND_IRQ	6	Stop until event conditions met AND an interrupt controller interrupt occurs

The rest of the fields define which conditions will be taken into account to restart a halted processor.

There are two types of events:

- Counted events: These decrement the field called ZERO. Flow controller resumes when this counter reaches 0
- Activity events: The flow controller resumes when the activity occurs, no counting

All conditions are disabled at reset:

- JTAG: Resume on JTAG activity
- SCLK : Resume on Nth SYSCLK cycle ticks
- X32K : Resume on Nth X32K clock input ticks
- uSEC: Resume on Nth uSEC clock ticks (uSEC = microsecond)
- mSEC: Resume on Nth mSEC clock ticks (mSEC = Millisecond)
- SEC : Resume on Nth second RTC clock ticks
- X_RDY: Resume on Nth XIO.RDY Ext. IO Ready events
- SMP3[1,0]: Resume on Nth SMP.3[1,0] Semaphore set events
- XRQ_[D,C,B,A] : Resume on Nth XRQ.[D,C,B,A] External Trigger events
- [O,I]B[E,F]: Resume on Nth [O,I]B[E,F] [Outbox, Inbox] [Empty, Full] Events
- [IRQ,FIQ]_[1,0]: Resume on [IRQ,FIQ].[1,0] IRQ Valid, 1 is COP, 0 is CPU
- ZERO: Initialized then decremented

Note: If more than one event is enabled, the event counter will decrement based on an or condition of enabled events

uSEC, mSEC and SEC are based on free running clocks, so there is a -1 to 0 period possible difference between the programmed delay value and the really observed delay value.

13.1.1 FLOW_CTLR_HALT_CPU_EVENTS_0

Offset: 000h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:29	0x0	MODE: 0 = FLOW_MODE_NONE 1 = FLOW_MODE_RUN_AND_INT 2 = FLOW_MODE_WAITEVENT 3 = FLOW_MODE_WAITEVENT_AND_INT 4 = FLOW_MODE_STOP_UNTIL_IRQ 5 = FLOW_MODE_STOP_UNTIL_IRQ_AND_INT 6 = FLOW_MODE_STOP_UNTIL_EVENT_AND_IRQ 2 = FLOW_MODE_STOP 3 = FLOW_MODE_STOP_AND_INT 4 = FLOW_MODE_STOP_UNTIL_INT 5 = FLOW_MODE_STOP_UNTIL_INT_AND_INT 6 = FLOW_MODE_STOP_OR_INT
28	0x0	JTAG
27	0x0	SCLK
26	0x0	X32K
25	0x0	uSEC
24	0x0	MSEC
23	0x0	SEC
22	0x0	X_RDY

Bit	Reset	Description
21	0x0	SMP31
20	0x0	SMP30
19	0x0	XRQ_D
18	0x0	XRQ_C
17	0x0	XRQ_B
16	0x0	XRQ_A
15	0x0	OBE
14	0x0	OBF
13	0x0	IBE
12	0x0	IBF
11	0x0	IRQ_1
10	0x0	IRQ_0
9	0x0	FIQ_1
8	0x0	FIQ_0
7:0	0x0	ZERO

13.1.2 FLOW_CTLR_HALT_COP_EVENTS_0

Offset: 004h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:29	0x0	MODE: 0 = FLOW_MODE_NONE 1 = FLOW_MODE_RUN_AND_INT 2 = FLOW_MODE_WAITEVENT 3 = FLOW_MODE_WAITEVENT_AND_INT 4 = FLOW_MODE_STOP_UNTIL_IRQ 5 = FLOW_MODE_STOP_UNTIL_IRQ_AND_INT 6 = FLOW_MODE_STOP_UNTIL_EVENT_AND_IRQ 2 = FLOW_MODE_STOP 3 = FLOW_MODE_STOP_AND_INT 4 = FLOW_MODE_STOP_UNTIL_INT 5 = FLOW_MODE_STOP_UNTIL_INT_AND_INT 6 = FLOW_MODE_STOP_OR_INT
28	0x0	JTAG
27	0x0	SCLK
26	0x0	X32K
25	0x0	uSEC
24	0x0	MSEC
23	0x0	SEC
22	0x0	X_RDY
21	0x0	SMP31

Bit	Reset	Description
20	0x0	SMP30
19	0x0	XRQ_D
18	0x0	XRQ_C
17	0x0	XRQ_B
16	0x0	XRQ_A
15	0x0	OBE
14	0x0	OBF
13	0x0	IBE
12	0x0	IBF
11	0x0	IRQ_1
10	0x0	IRQ_0
9	0x0	FIQ_1
8	0x0	FIQ_0
7:0	0x0	ZERO

13.1.3 FLOW_CTLR_CPU_CSR_0

The CPU_CSR registers are replicated for CPU0 and CPU1. 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.

> **Note:** The difference between LP1 and LP0 is not visible at the flow controller, but is defined by a configuration bit in PMC (LP1 entry triggers LP0 entry).

All fields of the CPU_CSR register have an initial value of 0. The different fields are:

- PWR_STATE : Current state of the powerGate State Machine
- WAIT_EVENT: CPU is waiting, wake up is via an event
- HALT: CPU is halted
- P2F_ACK: pmc2flow_ack signal, this is the same signal for both CPU cores
- F2P_PWRUP : flow2pmc_pwrup, this is the same signal for both CPU cores
- F2P_REQ : flow2pmc_req valid, this is the same signal for both CPU cores
- F2C_MPCORE_RST : TRUE when Requesting Reset of MPCore
- PWR_OFF_STS : TRUE when CPU PowerGated OFF by Flow Controller
- INTR_FLAG : TRUE when Interrupt is Active -- Write-1-to-Clear
- EVENT_FLAG : TRUE when Event is Active -- Write-1-to Clear
- WAIT_WFE_BITMAP : All cores indicated in bitmap must be in STANDBY_WFE before CPU PowerGating
- EVENT_ENABLE : Generates an event when the flow controller exits the halted state
- ENABLE: PowerGate Enable - Halt or Event-wait causes CPU PowerGating

Offset: 008h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxx00xxxxxxxx00xx00

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
5:4	RW	0x0	WAIT_WFE_BITMAP
1	RW	0x0	EVENT_ENABLE
0	RW	0x0	ENABLE

13.1.4 FLOW_CTLR_COP_CSR_0

COP Control/Status for Interrupts

R/W addr=6000:700c

Offset: 00ch | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
15	0x0	INTR_FLAG: TRUE when Interrupt is Active -- Write-1-to-Clear

13.1.5 FLOW_CTLR_XRQ_EVENTS_0

XRQ Event Detect Selector Register

Offset: 010h | Read/Write: R/W | Reset: 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.

Bit	Reset	Description
		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.

13.1.6 FLOW_CTLR_HALT_CPU1_EVENTS_0

Offset: 014h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:29	0x0	MODE: 0 = FLOW_MODE_NONE 1 = FLOW_MODE_RUN_AND_INT 2 = FLOW_MODE_WAITEVENT 3 = FLOW_MODE_WAITEVENT_AND_INT 4 = FLOW_MODE_STOP_UNTIL_IRQ 5 = FLOW_MODE_STOP_UNTIL_IRQ_AND_INT 6 = FLOW_MODE_STOP_UNTIL_EVENT_AND_IRQ 2 = FLOW_MODE_STOP 3 = FLOW_MODE_STOP_AND_INT 4 = FLOW_MODE_STOP_UNTIL_INT 5 = FLOW_MODE_STOP_UNTIL_INT_AND_INT 6 = FLOW_MODE_STOP_OR_INT
28	0x0	JTAG
27	0x0	SCLK
26	0x0	X32K
25	0x0	uSEC
24	0x0	MSEC
23	0x0	SEC
22	0x0	X_RDY
21	0x0	SMP31
20	0x0	SMP30
19	0x0	XRQ_D
18	0x0	XRQ_C
17	0x0	XRQ_B
16	0x0	XRQ_A
15	0x0	OBE
14	0x0	OBF
13	0x0	IBE
12	0x0	IBF
11	0x0	IRQ_1
10	0x0	IRQ_0
9	0x0	FIQ_1
8	0x0	FIQ_0

Bit	Reset	Description
7:0	0x0	ZERO

13.1.7 FLOW_CTLR_CPU1_CSR_0

Offset: 018h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxx00xxxxxxx00xx00

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
5:4	RW	0x0	WAIT_WFE_BITMAP
1	RW	0x0	EVENT_ENABLE
0	RW	0x0	ENABLE

14.0 LEVEL 2 CACHE CONTROLLER

The Tegra® 2 CPU complex contains dual ARM® Cortex-A9 CPUs and a PL310 L2 Cache controller. This section describes the L2 cache controller.

The L2 Cache Controller is an ARM PL310 L2 controller configured with 1 MByte of cache RAM.

This section does not document the PL310 itself; for that you will need the ARM documentation which is available from ARM. At the time of writing the latest version is:

PrimeCell Level 2 Cache Controller (PL310)
Revision: r2p0
Technical Reference Manual

Published by ARM Limited, document number ARM DDI 0246C.

Note: Current version at this time. You should periodically look for updates and refer to the latest available version of this material.

Implementation Details

Some aspects of the PL310 may be configured for a specific implementation. The Tegra 2 Series implementation has the following features:

- r2p0 revision level
- 8-way set-associative
- 1 Megabyte cache size
- Support for locking per master ID

15.0 MEMORY CONTROLLER

The memory interface subsystem of the Tegra® 2 Processor consists of two modules:

- **The Memory Controller (MC)**:
  The MC merges request streams from various client interfaces into request stream(s) for the various memory target devices, and returns response data to the various clients. The MC has a configurable arbitration algorithm to allow the user to fine-tune performance among the various clients.
- **The External Memory Controller (EMC)**:
  The EMC interfaces with the off-chip SDRAM to service the request stream sent from MC. The EMC also has various performance-affecting settings beyond the obvious SDRAM configuration parameters and initialization settings. Tegra 2 design supports multiple JEDEC standard protocols: LPDDR, LPDDR2, and DDR2.

15.1 Usage Modes

The various registers in MC and EMC can be summarized into various usage-mode categories for easy reference. The categories are:

- Config (Reserved) <Secure>: A configuration register, or if in parenthesis, a configuration setting reserved for special case purposes. Generally this is a timing or functionality configuration that is not strictly performance optimization, and will not normally be used. If in angle brackets, the register may be secured from untrusted accesses using ARM® TrustZone™ security protocol.
- Trigger: This register triggers an operation. There may be side-effects or an explicit programming sequence required.
- Perf Config: Configuration specific to performance. Should not cripple functionality even when not set to optimal.
- Debug: Useful for debug purposes.
- RO: Read-only. If in brackets, a write will trigger some sort of clear action.

The tables use * as a wildcard to group various registers of similar functionality.

Table 48 Register Usage Modes

Register	Config RSVD	Perf Config	Debug	Trigger	RO
Memory Controller					
INTSTATUS					[X]
INTMASK	X				
GART_ERROR_* DECERR_AXI_* DECERR_EMEM_OTHERS_* SECURITY_VIOLATION_*			X		X
EMEM_CFG EMEM_ADR_CFG	X				
EMEM_ARB_CFG*		X			
GART_CONFIG	X				
GART_ENTRY_ADDR	X				
GART_ENTRY_DATA	X			X	
TIMEOUT_CTRL		X			
CLKEN_OVERRIDE	(X)				
SECURITY_CFG*	<X>				
STAT_CONTROL			X	X	
STAT_STATUS STAT_EMC_ADDR_LOW STAT_EMC_ADDR_HIGH STAT_EMC_CLOCK_LIMIT STAT_EMC_CONTROL_* STAT_EMC_HIST_LIMIT_*			X		
STAT_EMC_CLOCKS STAT_EMC_COUNT_* STAT_EMC_HIST_*			X		X
CLIENT_CTRL	(X)				
CLIENT_HOTRESETN	(X)			X	
*_ORRC					X

Register	Config RSVD	Perf Config	Debug	Trigger	RO
FPRI_CTRL_* TIMEOUT_* TIMEOUT_RCOAL_* BWSHARE_EMEM_CTRL_* BWSHARE_* BWSHARE_TMVAL		X			
AXI_DECERR_OVR LOWLATENCY_CONFIG	(X)				
LOWLATENCY_RAWLOGIC_WRITE_ PARTICIPANTS		X			
CLIENT_ACTIVITY_MONITOR_ *			X		
External Memory Controller					
INTSTATUS					[X]
INTMASK	X				
DBG	(X)		X		
CFG		X			
ADR_CFG*	X				
REFCTRL	X				
PIN	X			X	
TIMING_CONTROL	X			X	
RC, RFC, RAS, RP, R2W, W2R, R2P, W2P, RD_RCD, WR_RCD, RRD, REXT, WDV, QUSE, QRST, QSAFE, RDV, REFRESH, BURST_REFRESH_NUM, PDEX2RD, PCHG2PDEN, ACT2PDEN, AR2PDEN, RW2PDEN, TXSR, TCKE, TFAW, TRPAB, TCLKSTABLE, TCLKSTOP, TREFBW, QUSE_EXTRA, ODT_WRITE, ODT_READ	X				
MRS, EMRS, REF, PRE, NOP SELF_REF, DPD, MRW, MRR	X			X	
CMDQ		X			
FBIO_*	X				
DQS_TRIMMER_RD*	X				[X]

Register	Config RSVD	Perf Config	Debug	Trigger	RO
CLKEN_OVERRIDE	(X)				
LL_ARB_CONFIG T_MIN_CRITICAL_HP T_MIN_CRITICAL_TIMEOUT T_MIN_LOAD T_MAX_CRITICAL_HP T_MAX_CRITICAL_TIMEOUT T_MAX_LOAD		X			
STAT_CONTROL			X	X	
STAT_STATUS STAT_PWR_CLOCK_LIMIT STAT_LLMC_ADDR_LOW STAT_LLMC_ADDR_HIGH STAT_LLMC_CLOCK_LIMIT STAT_LLMC_CONTROL_0 STAT_LLMC_HIST_LIMIT_0			X		
STAT_PWR_CLOCKS STAT_PWR_COUNT STAT_LLMC_CLOCKS STAT_LLMC_COUNT_0 STAT_LLMC_HIST_0 STAT_DRAM_CLOCK_LIMIT_* STAT_DRAM_CLOCKS_* STAT_DRAM_DEV*_*_CNT_* STAT_DRAM_DEV*_CUMM_BANKS_ACTIVE_CKE_EQ*_* STAT_DRAM_DEV*_CKE_EQ1_CLKS_* STAT_DRAM_DEV*_EXTCLKS_CKE_EQ*_*			X		X
AUTO_CAL_CONFIG	X			X	
AUTO_CAL_INTERVAL	X				
AUTO_CAL_STATUS					X
REQ_CTRL	X				
EMC_STATUS					X
CFG_2	X				
CFG_DIG_DLL	X	X			X
DLL_XFORM_DQS	X				
DLL_XFORM_QUSE	X				

Register	Config RSVD	Perf Config	Debug	Trigger	RO
DIG_DLL_UPPER_STATUS					X
DIG_DLL_LOWER_STATUS					X
CFG_CLKTRIM_*	X				
CTT_TERM_CTRL	X				X
ZCAL_*	X				

The MCCIFs (client interfaces) also have per-client configuration settings. These registers are grouped with the requesting module and not in MC. "m" here stands for various module names, "c" stands for the client name, and "?" and "*" stand for wildcards. A few modules might not use the provided register instantiation script and their naming conventions may vary slightly (e.g. display)

Table 49 Client Interfaces Configuration Settings

Register	Rsvd Config	Perf Config	Debug	Trigger	RO
MCCIF Register Fields					
m_MCCIF_FIFOCTRL. m_MCCIF_RDCL_RDFAST m_MCCIF_WRMC_CLLE2X m_MCCIF_RDMC_RDFAST m_MCCIF_WRCL_MCLE2X		X			
m_TIMEOUT_WCOAL_m. c_WCOAL_TMVAL		X			
m_MCCIF_c_HP.c2MC_HPTM m_MCCIF_c_HP.c2MC_HPTH		X			
m_MCCIF_c_HYST. c_HYST_EN c_HYST_REQ_TH c_HYST_TM c_DHYST_TH c_DHYST_TM c_HYST_REQ_TM		X			

15.2 Programming Sequence

15.2.1 Minimal Initialization

The minimal sequence required to start accessing SDRAM memory is:

1. Program any necessary static pad configuration
2. Bring MC/EMC out of reset and turn on clocks
3. Program static configuration registers:
   MC_EMEM_CFG, MC_EMEM_ADR_CFG, EMC_ADR_CFG,
   MC_LOWLATENCY_CONFIG.LL_DRAM_INTERLEAVE, EMC_RC (and other DRAM timing registers)
   EMC_FBIO_CFG1, EMC_FBIO_CFG5, EMC_FBIO_CFG6, EMC_FBIO_DQSIB_DLY, EMC_FBIO_QUSE_DLY.
   The exact settings and meaning of these registers are described in the registers section below. To help translate a SDRAM vendor device spec into these parameters, please contact your NVIDIA FAE to obtain the spreadsheet used to compute register settings from the DRAM specification.
4. After the static configuration register values have been updated, you must commit the values by moving them in front of the shadow register set to active register via: RegWrite(EMC_TIMING_CONTROL, 0x1)
5. Initialize the SDRAM. This sequence may vary slightly depending on the requirements of a particular device. Tegra 2 devices support different protocols (LPDDR2, DDR2) as well as vendor-specific timings for certain models. For example, the DDR2 sequence will look something like this:
   a. Apply power to the device.
   b. Wait until voltage is stable.
   c. Apply stable clocks.
   d. Wait 200 µs .
   e. Assert and hold CKE high: RegWrite(EMC_PIN_0.PIN_CKE, NORMAL)
   f. Precharge all banks: RegWrite(EMC_PRE_0, 1)
   g. Auto refresh × 2: RegWrite(EMC_REF_0, 2<<1+1)
   h. Extended mode register set: RegWrite(EMC_EMRS_0, (see [4]))
   i. Mode register set: RegWrite(EMC_MRS_0, (see [4]))
   j. Enable hardware-inserted refreshes: RegWrite(EMC_REFCTRL_0, ENABLED)

After this initialization sequence is complete, various other sequences can occur, such as GART initialization, performance parameter tuning, power-down, etc.

The optimal settings for EMC_FBIO_DQSIB_DLY, EMC_FBIO_QUSE_DLY, and EMC_FBIO_CFG6 etc are obtained by a characterization process. Contact your NVIDIA applications engineer for more details of this process.

15.2.2 GART

The GART (Graphics Address Relocation Table) allows MC to provide a linear view of memory that has been fragmented. GART table entries are programmed using an indirect mechanism. An address register provides the offset of the GART entry to be accessed. Reading or writing a data register causes the GART entry to be read or written. The GART is disabled at reset and must be initialized before it can be enabled. To initialize the GART, all entries must be marked either INVALID or VALID.

To initialize the GART to all-INVALID, do:

```
for (uint i = 0; i < (GART_SIZE); i += (GART_PAGE_SIZE)) {
        RegWrite(MC_GART_ENTRY_ADDR_0, i);
        RegWrite(MC_GART_ENTRY_DATA_0, 0x0);
}
```

Where GART_SIZE is 32 MB and GART_PAGE_SIZE is 4 KB.

After the GART table has been initialized (either to all-INVALID or to the desired mapping), it may be enabled:

```
RegWrite(MC_GART_CONFIG_0, ENABLE);
```

To configure a GART entry K to point to EMEM address J, do:

```
RegWrite(MC_GART_ENTRY_ADDR_0, K);
RegWrite(MC_GART_ENTRY_DATA_0,   J | 1<< GART_ENTRY_DATA_PHYS_ADDR_VALID_SHIFT);
```

Setting the topmost bit of the address to 1 is what marks the GART entry as valid.

15.2.2.1 GART Debugging Comments

If a client attempts to access any GART-aperture address while the GART is disabled, or if the EMEM address encoded by a valid GART entry is not within the range defined by MC_EMEM_CFG.EMEM_SIZE_KB, a DECERR_EMEM_OTHERS interrupt will be triggered. If a client attempts to access a GART address that is marked INVALID, the INVALID_GART_PAGE interrupt will be triggered. If the EMEM address encoded by a valid GART entry is in the secured aperture of EMEM, and if the request is not marked secure, the SECURITY_VIOLATION interrupt will be triggered. All these interrupts log information about the error in their corresponding register field.

GART entries may be programmed at any time; however if a GART entry is being used by a client at the same time it is being programmed, the behavior is undefined because the ordering of the client access and the register write cannot be guaranteed.

15.2.3 EMC Dynamic Clock Change

Maintaining the maximum frequency for accessing DRAM can cause unnecessarily large power consumption in some applications. While 3D applications need the maximum memory bandwidth, low-power audio playback may need just minimum bandwidth. The complex trade-off between bandwidth and power should be discussed with your NVIDIA FAE to find the latest recommendations.

Changing clock frequency or voltage while the SDRAM is running requires careful sequencing of steps to ensure that no timing requirements are violated in the process. To provide clock frequency and voltage scaling flexibility without interruption of normal memory operations, the EMC implements logic for glitch-less transition of configuration setting through the use of shadow registers and a hardware handshake with the clock-source unit (CAR).

A clock change that includes reprogramming the DRAM mode registers atomically within that clock change is not supported on Tegra 2 devices; therefore, the user cannot change the timing parameters such as CAS latency (or Read Latency for LPDDR2) and tWR with clock change. Instead, the mode registers should be programmed only at boot time with worst case setting. For example, if in 200 MHz operation the CAS latency could be set to 3 while for 333 MHz the CAS latency must be 5, then the user must always set the CAS latency to 5.

There are two modes for frequency change, power-down mode and self-refresh mode, depending on the mode EMC puts DRAM into during frequency change. DDR2 can only use self-refresh mode, since it will otherwise require a DLL reset through mode register. LPDDR2 can work on either mode, but power-down mode is recommended.

If both PLL source and clock divider need to be changed, it must be done in two separate steps at least 1 micro second apart. MC and EMC will run at an intermediate clock frequency between those two steps, so the first step must program the EMC timing registers for the intermediate frequency while the second step must program for the final frequency. Also note that the intermediate frequency must be chosen not to exceed the valid frequency range for the DRAM, described below.

Please contact an NVIDIA FAE for the supported clock frequencies and timing and trimmer register programming at those frequencies.

Clock change sequence is the same for both low to high and high to low frequency change.

Clock Change Procedure

1. At boot time, choose the right frequency change mode for the DRAM:

- DDR2:
  EMC_CFG_2.CLKCHANGE_PD_ENABLE=0
  EMC_CFG_2.CLKCHANGE_SR_ENABLE=1
- LPDDR2:
  EMC_CFG_2.CLKCHANGE_PD_ENABLE=1 (reset value)
  EMC_CFG_2.CLKCHANGE_SR_ENABLE=0 (reset value)

2. At the time of clock change:
   - Software programs shadowed timing registers and trimmer registers to the target frequency and voltage.
   - Program CAR registers to change PLL source or clock divider (but not both – see above). This action automatically triggers the handshake between CAR and EMC and the update of the active EMC registers from the shadowed set.
   - Keep a 1us gap before the next clock change.

15.3 Performance Configuration Registers

Arbitration tuning is closely tied to the hardware, changes to the default NVIDIA software settings should not be made without extensive testing over multiple use-cases. Consult your NVIDIA FAE before attempting to modify the default settings.

15.3.1 Arbitration Overview

It is the job of MC to merge request streams from various client interfaces into request stream(s) for the SDRAM memory. To accomplish this goal, each module (major units like 2D, 3D, VDE etc) has several client interfaces to service the different data streams the module requires; since read and write requests require separate client interfaces, the simplest modules generally have one read and one write client. Each client's stream is scheduled independently by the MC; very little pre-arbitration is done in the modules themselves. Only the head of a client's un-arbitrated request stream is considered for arbitration due to implementation complexity limitations.

For coherency purposes, the arbiter is considered the point of coherency where the read/write request ordering is fixed. This imposes the restriction on EMC to not reorder the request stream, and so this implies that the exact ordering of the request stream from MC should be scheduled to efficiently use the external SDRAM. The various characteristics of SDRAM scheduling are discussed in many external sources; most important to the MC arbitration algorithm are page-miss penalties.

To calculate the maximum bandwidth available for a SDRAM configuration:

clock frequency * (SDRAM device data bus width == 32 ? 8 : 4)

Some examples:

400MHz x32 DDR2: 3.2GB/s

300MHz x32 LPDDR2: 2.4 GB/s

The most direct measurement of bandwidth efficiency is the number of used cycles on the SDRAM data bus; the page close and reopen command sequences must be scheduled efficiently to avoid unused cycles on the SDRAM data bus. Different mixes of requests and clients can lead to drastically different bandwidth efficiencies. For example:

- A read client that accesses sequential memory locations (linear client) can achieve 98% of total bandwidth
- Mix several linear read clients (or several linear write clients) and you still get about 98%
- Mix read with write (linear clients) then you drop to about 80 - 90% (80% for 1R/1W, 90% for all clients)
- If you have multiple clients accessing "random" addresses (no page reuse), then you drop to 30 - 60% depending on the number of active clients (30% for 1 client, 60% for all clients)

When planning for use-cases, the type of streams required - including background traffic - should be taken into account and used to estimate the expected bandwidth efficiency.

The MC has optimized for SDRAM bandwidth over latency considerations. In Tegra 2 devices, the presence of a high-performance ARM processor complex suggests at least the CPU memory clients must be optimized for latency, and that some arbitration knobs are required to balance between the bandwidth and latency optimizations.

Aside from the algorithm sections to shape the stream efficiently for the target memory device, the MC also implements algorithms for fair scheduling and starvation avoidance amongst the clients, and prioritization when the demand exceeds the available bandwidth. Each client has a timeout counter (to prevent starvation), a bandwidth share credit counter (for guaranteed bandwidth allocations), and a programmable "fixed" priority. Some real-time clients have a high-priority signal generated by comparing the client interface FIFO level with a programmable threshold to further limit starvation.

During an arbitration cycle, the arbiter first culls from the list any requests that would cause unnecessary bank thrashing (relative to a threshold counter). Then, it feeds the remaining requests into two binary competition trees, one for read requests and one for write requests. In each tree, the request with the highest priority vector wins. The priority vector is calculated as:

```
// define hp : high priority signal from client
// define tm : time-out counter expired
// define bw : bandwidth share credits available
// define fpri[1:0]: client fixed priority from register
// define sp : access would not produce a page miss
// priority = {(hp ? 3'h7 : {tm,bw} + fpri[1:0]), sp}
```

Finally, the arbiter arbitrates between the read and write winners. The read/write arbitration takes into consideration the priority vector while attempting to minimize the read/write switches on the DRAM interface.

15.3.1.1 Page-miss Efficiency

Page-miss efficiency is affected by 3 classes of arbitration knobs: those that control the MC's stream shaping, those that configure the dynamic auto pre-charge, and those that control EMC's command-queue look-ahead search.

MC Stream Shaping

The register fields discussed in this section are:

- MC_EMEM_ARB_CFG2.EMEM_BANKCNT_RD_TH
- MC_EMEM_ARB_CFG2.EMEM_BANKCNT_WR_TH
- MC_EMEM_ARB_CFG2.EMEM_BANKCNT_NSP_RD_TH
- MC_EMEM_ARB_CFG2.EMEM_BANKCNT_NSP_WR_TH
- MC_EMEM_ARB_CFG1.EMEM_SP_MAX_GRANT
- MC_EMEM_ARB_CFG0.EMEM_SP_MAX_GRANT_OVERALL

There are 4 threshold per-bank counter controls in MC that can affect the stream going to EMC. All of these are used to block requests known to cause a possible page-miss in the SDRAM for a number of cycles. Due to implementation/complexity limitations, the MC arbiter attempts to guess which sequences would produce page-misses based on a per-client same-page indication. Requests signaling same-page are known to not cause a page-miss with respect to the previous request from the same client, and all clients are assumed to be working on different pages.

The BANKCNT_*_TH setting is used by the MC arbiter to encourage bank rotation among likely page-misses while taking advantage of page-hits from individual clients; this mimics the minimum page-miss cycle time parameter in SDRAM. The BANKCNT_NSP_*_TH setting is used by MC arbiter to extend the block-out for known page-misses generated by the client streams; in other words, it is penalizing clients that cause bank-thrashing.

The RD version of these thresholds are used if the access that started the counters was a read, otherwise the WR version is used. In general, the thresholds should be set to mimic the page-miss delays experienced by the external memory device but on the MC clock: BANKCNT_*_TH accounts for tRAS while BANKCNT_NSP_*_TH accounts for tRAS+tRP(+tWR if LPDDR2

writes). Thresholds that are too large will lead to wasted databus cycles on the SDRAM while thresholds that are too small will lead to EMC back pressuring MC.

Example settings are dependent on the MC/EMC clock-frequency and the DRAM parameters, as shown (all units in cycles):

Table 50. BANKCNT_NSP_TH Settings

Config	tRAS	tRP	tWR	RD_TH	NSP_RD_TH	WR_TH	NSP_WR_TH
533MHz DDR2	22	7	2	16	27	27	27
400MHz LPDDR2	17	8	3	TBD	TBD	TBD	TBD
333MHz LPDDR2	14	6	3	13	18	21	18

The EMEM_SP_MAX_GRANT is designed to combat a starvation side-effect of the tiling implementation for Tegra 2 devices. Once a particular client has opened a page, as long as that client's subsequent requests have the same-page indication set and if the client maintains a request rate faster than the bank threshold countdown, all other clients trying to open a new page will essentially be blocked from arbitration by the bank thresholds. To keep this behavior from producing starvation in the other clients, the arbiter forces off the winning client's same-page indication after MAX_GRANT winning requests to a single bank when bank contention is detected. This allows the bank thresholds to count down and all clients to compete at the next available arbitration cycle. A value of 0x10 approximates previous chip’s performance.

The EMEM_SP_MAX_GRANT_OVERALL is designed to break starvation effects where multiple clients are blocked in a SP_MAX_GRANT scenario but bank contention is not occurring. After MAX_GRANT+OVERALL winning requests, the winning client’s same-page indication is forced off for one arbitration cycle to allow all clients to compete during that arbitration cycle. To prevent interference with MAX_GRANT masking, the value of OVERALL should be > 1.

EMC Command Look-ahead

The register fields discussed in this section are:

- EMC_CMDQ.PRE_DEPTH
- EMC_CMDQ.ACT_DEPTH
- EMC_CMDQ.RW_DEPTH
- EMC_CMDQ.RW_WD_DEPTH

The EMC maintains a FIFO of un-serviced requests, and peeks into this FIFO to generate FIFOs of future page reopen commands. All of these FIFOs have programmable depths. In general, a deeper FIFO improves bandwidth utilization at the cost of slower response time for a latency-sensitive request. The RW depth must be greater than the ACT and PRE depths. The default values (pre,act,rw, wd=4,4,8,16) are the best known combination for LPDDR2 and DDR2 devices at POR clock speeds.

15.3.1.2 Read/Write Arbitration

The register fields discussed in this section are:

- MC_EMEM_ARB_CFG0.EMEM_WRCNT_TH
- MC_EMEM_ARB_CFG0.EMEM_RWCNT_TH
- MC_EMEM_ARB_CFG1.EMEM_RCL_MASK
- MC_EMEM_ARB_CFG1.EMEM_WCL_MASK
- MC_EMEM_ARB_CFG1.EMEM_NORWSWITCH_BKBLOCK
- MC_EMEM_ARB_CFG1.EMEM_NOWRSWITCH_BKBLOCK
- MC_EMEM_ARB_CFG1.EMEM_RWSWITCH_RWINEXPIRED
- MC_EMEM_ARB_CFG1.EMEM_WRSWITCH_WWINEXPIRED

- MC_EMEM_ARB_CFG1.EMEM_SP_MAX_GRANT_RW

The write/read switch is used to make the arbiter switch from writes to reads once the write window cycle count has expired and there are read and write requests with equal priority. If the count has not expired or if the write winner's priority is greater than the read winner's priority, the arbiter will stick with writes as it is better for SDRAM bandwidth utilization. Switching is usually the desired behavior when priorities are equal.

The WRCNT_TH sets the threshold for the number of writes in a window before the arbiter switches to preferring reads; similarly RWCNT_TH sets the number of reads before switching to writes. The priority at the RW arbiter is determined by the winning client's priority vector plus an LSB that encodes whether the read/write window has expired for the respective transaction. The bits of the priority vector are rearranged such that the same-page bit moves to bit [1] from bit [0], effectively prioritizing same-page efficiency over the static arbitration due to fixed priority and bandwidth share. The same-page bit is masked by the status of the SP_MAX_GRANT_RW threshold, such that after a certain number of winning requests, the same-page indication is no longer considered as part of the RW arbiter's priority.

The priority comparison can be modified by the MASK fields if desired. The SWITCH_*WINEXPIRED fields can be used to force a switch on window expiration. The NO*SWITCH_BKBLOCK fields can be used to avoid a switch when there are requests pending for the current window that are waiting for a BANKCNT timer to expire.

The bandwidth efficiency for linear as well as random traffic improves with larger window sizes. It also improves when the window is not forced to switch when it expires and when is prevented from switching on a blocked bank.

15.3.2 Priority Vector Bits

15.3.2.1 High-Priority Threshold

- m_MCCIF_c_HP.c2MC_HPTM
- m_MCCIF_c_HP.c2MC_HPTH

The high-priority should be enabled for hard real-time clients only. The values to program depend on the client bandwidth requirement and the client versus memory controller's clock ratio. Even if a client is requesting HP, it still can get undesirable service levels due multiple clients requesting HP; also if the client itself produces a request stream with page-misses, this can cause other traffic to sneak in.

For reads, the high-priority is set if the number of entries in the return data FIFO is under the threshold (HPTH). The high-priority assertion can be delayed by a number of memory clock cycles indicated by the timer (HPTM). This combination of threshold and timer creates a hysteresis effect, avoiding setting the high-priority for very short periods of time, which may or may not be desirable.

For writes, the high-priority is set if the number of entries in the data FIFO is over the threshold (HPTH).

The CPU's low-latency path arbitration includes a factor to account for the number of clients requesting high-priority; thus spurious high-priority requests should be avoided.

The Display clients have an additional HP suppression feature to prevent the Display clients from producing spurious high-priority requests when at start/end of frame or when the data FIFO fill rate drops slightly below the consumption rate (but does not threaten underflow). The Display controller provides the MCCIF with a start-of-frame indication, the consumption rate and the time available before the first word is needed. The MCCIF then combines this information with a current return-data FIFO fill level to decide whether to suppress the basic fill-level HP assertion. This feature does not have any MCCIF-related register programming but is mentioned here for completeness.

15.3.2.2 Timeout

The register fields discussed in this section are:

- MC_TIMEOUT_CTRL.EMEM_TM_SFACTOR

- MC_TIMEOUT_CTRL.TMCREDITS
- MC_TIMEOUT_*.*_TMVAL

If a client has requests pending but not arbitrated for more than EMEM_TM_SFACTOR * TMVAL cycles, the timeout bit in the priority vector is set. When the timeout bit is set, if TMCREDITS is set to FROM_CIF_FIFO, the timeout bit will remain set until all pending requests at the time of the timeout are arbitrated. If TMCREDITS is set to ONE, then only one request will be arbitrated for each timeout.

Timeout can be used to protect against starvation for a low-priority client, but if a client times out often it has an adverse effect on bandwidth efficiency.

15.3.2.3 Bandwidth Share

The register fields discussed in this section are:

- MC_BWSHARE_TMVAL.BW_TM_SFACTOR1
- MC_BWSHARE_TMVAL.BW_TM_SFACTOR2
- MC_BWSHARE_EMEM_CTRL_*.*_BW_EMEM
- MC_BWSHARE_*.*_BW_TMSFACTORSEL
  MC_BWSHARE_*.*_BW_ALWAYSINC
- MC_BWSHARE_*.*_BW_MAXTH
- MC_BWSHARE_*.*_BW_HIGHTH
- MC_BWSHARE_*.*_BW_INCVAL

Bandwidth sharing is a way to specify guaranteed bandwidth for a client. Bandwidth share is implemented using a token-bucket algorithm, with one bucket per module. A programmable amount of credits (INCVAL) are added to the bucket on update pulses and removed when a request is granted; the credit counter saturates at programmable maximum threshold (MAXTH). When the credit level exceeds a programmable high threshold (HIGHTH), the bandwidth share priority vector bit for the client is asserted; when the credit level reaches zero, the bandwidth share priority vector bit for the client is de-asserted. The bandwidth-share caps the credit counter to a maximum value with MAXTH. This is called “use it or lose it”; once the counter reach the maximum, it saturates and so “loses” the credit associated with new updates. The exact value of MAXTH controls the maximum size of a high priority burst. The units of INCVAL are bytes. The units of MAXTH and HIGHTH are in terms of memory requests, which are 16 bytes.

A module must choose to participate in the bandwidth share algorithm by programming the associated MC_BWSHARE_<module> register and enabling the appropriate MC_BWSHARE_EMEM_CTRL_*.<module>_BW_EMEM bit. In addition to the basic token-bucket credit parameters, each module may choose to increment credits only when requests are pending arbitration, or all the time (ALWAYSINC). There are two update pulses in the bandwidth share logic; each module may choose which to use (TMSFACTORSEL). The frequency of update pulses syncs is programmable via the BW_TM_SFACTOR* fields.

The following rules should always be taken into account when programming the token bucket parameters:

- The reserved bandwidth should be less than the long term average bandwidth of the client, except for clients where the bandwidth bit is used to provide low latency. This ensures that the bandwidth-share bit eventually deasserts.
- The sum of reserved bandwidth should be significantly below the available bandwidth to ensure that the bandwidth-share bit will operate as expected.

Algorithm Details

The behavior of the algorithm is heavily influenced by the ratio between the update period and the average inter-arrival time for requests. When the update period is small compared to the time between bursts, the algorithm results in fine grain interleaving

of multiple streams at the burst level. When the update period is large it acts as a window protocol, possibly grouping multiple bursts, resulting in larger latency values.

The first important distinction is to distinguish between step and ramp operation. In step operations, INCVAL >= HIGHTH. This implies that the bandwidth-share bit is set after each update and because all clients are updated simultaneously, this results in synchronized operation of all clients (windowing). The choice between ramp and step operation is mostly a global choice as the value of INCVAL is applied to the client counters by the (globally defined) TM_SFACTOR* fields.

The second important distinction is between ALWAYSINC and only when requests are present. When ALWAYSINC is de-asserted, the operation is more akin to a timeout, but with a known amount of requests allowed at high priority once the HIGHTH limit is reached. The normal timeout bit drains the full request queue when it triggers and has gross time granularity.

A final distinction is for burst sources. They can be operated with a reserved bandwidth higher than their long term average, in which case the module operates normally at with its bandwidth-share bit asserted, and gets a decreased priority during long bursts. This is an operating mode that suits modules that are latency sensitive; they usually get low latency but are throttled down if they start requesting too much bandwidth.

A Simple Example

A low-priority module (M) need to maintain a certain minimum write bandwidth and is not latency-sensitive. First, solve:

target rate in bytes/sec = *i*/2^(*sf*+1) * mcclk frequency

where *i* is the number of bytes per update pulse and *sf* is the chosen scaling factor for one of the two update pulses. For a simple bandwidth requirement, either ramp (*i*<HIGHTH) or step (*i*>=HIGHTH) would be appropriate; if there are no other bandwidth-share clients to consider, save power with a larger *sf* and a HIGHTH *h*>*i* for step operation. Then, program MAXTH *x*>=*h*. MAXTH controls the length of the burst of prioritized requests the module can produce. Finally, program:

1. RegWrite(MC_BWSHARE_TMVAL, BW_TM_SFACTOR=sf)
2. RegWrite(MC_BWSHARE_<M>,
   <M>_BW_INCVAL = i
   <M>_BW_HIGHTH = h
   <M>_BW_MAXTH = x
   <M>_BW_ALWAYSINC = 0
   <M>_BW_TMSFACTORSEL=0)
3. RegWrite(MC_BWSHARE_EMEM_CTRL_0,
   <MREAD>_BW_EMEM = DISABLE)
4. RegWrite(MC_BWSHARE_EMEM_CTRL_1,
   <MWRITEY>_BW_EMEM = ENABLE
   <MWRITEU>_BW_EMEM = ENABLE
   <MWRITEV>_BW_EMEM = ENABLE)

A Complex Example

Assume 166MHz DDR2. Display Controller A is driving a HD panel with a single window at 1280x720, which is calculated to require 150MB/s. Display controller B is driving a WVGA panel at with a single window at 800x480, which requires 56MB/s, and that the protocol used to drive the WVGA panel causes a very bursty pattern of memory requests.

Assuming Display0A is used for driving 1280x720 panel and display1BB client is used for driving a 800x480 panel. The actual clients display would use for this usecase may vary.

Calculate INCVAL using the formula

BW required in bytes/sec = i/2^(sf+1) * mcclk frequency

Since INCVAL can only be integer value in bytes, it may not be possible to specify exact bytes/sec value for the update period. Pick scale factor and increment value pair to give almost the required bandwidth. (sf=1, i=4), (sf=2, i=8), (sf=3, i=16) would

almost give a 160MBytes of bandwidth, which is reasonable for Display controller A. Similarly for Display1BB client, selecting sf=3, i=6 would give around 60MBps.

If Display chooses to use clients with in same super clients, then the INCVAL should be sum of INCVAL values of both the clients. Also program BW_MAXTH_VAL to at least 4 or more. Requests equal to MAXTH_VAL are scheduled back to back (if available). Bigger number is preferred as it would reduce the disruption to the normal scheduling.

15.3.2.4 Fixed Priority

The register fields discussed in this section are:

- MC_FPRI_CTRL_*.*_PRIVAL

The fixed priority is exactly that; if two clients have different fixed priorities, then all other things being equal (timing, other priority vector bits, etc), the client with the higher priority will always win arbitration. The fixed-priority fields can be used to force certain clients to "soak up" any remaining bandwidth not required by more latency-sensitive clients. This can be accomplished by setting all units to LOW priority except the soakers, which are set to LOWEST. An example would be the 3D's FDC unit in regular use-cases. Fixed priority should be used sparingly because it clobbers some of the bank-efficiency arbitration when two clients do not have the same FPRI. Also, note that the differentiation between HIGH or MED and LOW* is much greater than the difference between HIGH and MED or LOW and LOWEST; this is due to the read-write winner arbitration prioritizing same-page over the least-significant bit of the priority vector.

15.3.3 Low-Latency

The registers discussed in this section are:

- MC_LOWLATENCY_RAWLOGIC_WRITE_PARTICIPANTS
- EMC_LL_ARB_CONFIG.MAX_LL_GREED
- EMC_LL_ARB_CONFIG.ALLOW_IDLE_INSERT
- EMC_LL_ARB_CONFIG.DIE_OFF_EXP
- EMC_LL_ARB_CONFIG.LL_GREED_DIFF_BANK_DISABLE
- EMC_LL_ARB_CONFIG.LL_GREED_DIFF_BANK_AFTER_DISABLE
- EMC_LL_ARB_CONFIG.LL_FORCE_INSERT_WR_DISABLE
- EMC_LL_ARB_CONFIG.LL_INSERT_DIFF_BANK_BEFORE_DISABLE
- EMC_LL_ARB_CONFIG.LL_INSERT_DIFF_BANK_AFTER_TOGGLE
- EMC_LL_ARB_CONFIG.LL_INSERT_DIFF_BANK_AFTER_REMOVE
- EMC_T_MIN_CRITICAL_HP
- EMC_T_MIN_CRITICAL_TIMEOUT
- EMC_T_MIN_LOAD
- EMC_T_MAX_CRITICAL_HP
- EMC_T_MAX_CRITICAL_TIMEOUT
- EMC_T_MAX_LOAD

15.3.3.1 Low-Latency Coherency

The architecture of the low-latency path creates a window of opportunity for a coherency error, where the write has been arbitrated, but a low-latency read slips into the stream before that write. To combat this, there is a coherency checker added to the EMC's low-latency logic; it blocks reads from occurring if there is a write in-flight to the same bank. This checker is conservative by default, as it checks all write clients against the low-latency paths. The checker may be made less conservative if a client's data stream is known to not interact with CPU data stream, the client's LL_RAW_PARTICIPATE bit may be disabled for slight improvements to the average CPU read latency.

15.3.3.2 Low-Latency Arbitration

The arbitration mechanism between the low latency path and the standard path incorporates a bandwidth-cap and minimal-impact insertion ideas to control the disruption due to low-latency on the standard path arbitration. The arbiter keeps a record of the system load, including system-stress indicators, and then attempts to insert the low-latency transaction into the standard stream at an opportune time between the current minimum and maximum delay suggested by the system load. A key guarantee is the latency of the low-latency path is never worse than the standard path latency. .

The exponent of the exponential-decay load-averaging equation can be modified with the DIE_OFF_EXP field; higher exponents increase the load-averaging window. The minimum/maximum insertion delays are calculated by:

lookup_load_delay(current_load_avg) + lookup_crit_delay (current_crit_count)

where the current system load indicators are used to index into tables of delay cycles programmed by the user in the T_MIN and T_MAX registers.

To improve the efficiency of the low-latency path, multiple transactions can be inserted back-to-back if they are to the same page; the number of transactions grouped in this way can be controlled by the MAX_LL_GREED field, however the current default of 3 is recommended for this architecture based on AXI behavior for ARM CPU cores. The best-case CPU latency in an idle system can be improved by switching on the ALLOW_IDLE_INSERT field; this is recommended because the minor increase in latency in any of the standard-path traffic should not be a problem in a lightly-loaded system.

The default low-latency arbitration configuration (except maybe for ALLOW_IDLE_INSERT) should be reasonable for most situations. In a case where linear reads and writes are occurring on the standard and low-latency paths, these settings result in a total bandwidth efficiency drop of less than 15%, with 20% of that traffic being low-latency requests. A more randomized scenario would find a lower efficiency drop, due to more opportunities for the low-latency logic to insert a transaction without disrupting the standard stream.

15.3.4 Miscellaneous Performance Knobs

15.3.4.1 Coalesce Timers

The register fields discussed in this section are:

- m_TIMEOUT_WCOAL_m.c_WCOAL_TMVAL
- MC_TIMEOUT_RCOAL_*.*_RCOAL_TMVAL
- MC_RCOAL_AUTODISABLE_*

These registers control the amount of time a request can be delayed in the coalesce buffers; this adds latency but improves bandwidth efficiency. Coalescing will only occur if two subsequent 16-byte requests to the same client are the first and second halves of a single 32-byte aligned aperture; the two 16-byte requests are merged into a single 32-byte request. By default, all write coalescing is turned on (TMVAL=50) and all read coalescing is turned on (generally TMVAL=4). .

The write coalescing time-out should be programmed depending on the client behavior. If the client has low locality or is sensitive to latency, then WCOAL should be turned off. If the client is not latency-sensitive, is known to have a linear-style access pattern, and/or if it has bursts of back-to-back linear accesses, it is recommended that the WCOAL value be programmed in a way that balances the latency increase and the bandwidth efficiency for the overall use-case.

The read coalescing time-out should be programmed depending on the client behavior and the client versus memory controller clock ratio. Because the read-coalescing FIFO resides in the MC, any calculations based on the client-side stream must be converted to memory controller clock. If the client has low locality or is sensitive to latency, then RCOAL should be turned off. If the client is not latency-sensitive, is known to have a linear-style access pattern, and/or if it has bursts of back-to-back linear accesses, it is recommended that the RCOAL value be programmed in a way that balances the latency increase and the bandwidth efficiency for the overall use-case.

Certain address sequences generated by tiled block read clients are known to be non-coalescable; to allow the user to program RCOAL for the more general case without suffering a latency penalty for these sequences, the read coalesce engine has an auto-disabling feature. When such a sequence is detected, the RCOAL will act as if it were disabled until a coalescable sequence is detected. This behavior is turned on by default, but can be disabled by writing the per-client AUTODISABLE_EN enable bit to DISABLED.

15.3.4.2 Client Hysteresis

The register fields discussed in this section are:

- m_MCCIF_c_HYST.c_HYST_EN
- m_MCCIF_c_HYST.c_HYST_REQ_TH
- m_MCCIF_c_HYST.c_HYST_TM
- m_MCCIF_c_HYST.c_DHYST_TH
- m_MCCIF_c_HYST.c_DHYST_TM
- m_MCCIF_c_HYST.c_HYST_REQ_TM

The effect of requests “dribbling” into the arbiter (whether due to isochronous request stream or simple clock-crossing latency) has a highly detrimental effect on arbiter efficiency. To combat this effect, clients known to behave in this manner will hold off sending requests until either a burst of requests is available or a timeout is reached. The burst-size required to maintain efficiency is 4 requests for most cases. This hysteresis-based grouping is sufficient to alleviate normal usecase strain on the arbiter due to isochronous and clock-crossing dribble.

A special case of hysteresis is required to improve the power profile of Display traffic to DRAM during the WinIdle usecase. Because the Display controller assumes it must cover a large average latency, it attempts to keep its return-data FIFO full at all times; also, Display consumes data at a much lower rate than the memory controller can produce the data. Thus in the WinIdle usecase, Display traffic prevents DRAM from entering its lowest-power state due to constant top-offs of the Display return-data FIFO. To counteract this tendency, an additional level of hysteresis, called Deep Hysteresis (DHYST), has been implemented for Display clients, with a burst-size threshold that defaults to half the return-data FIFO size.

Both the normal and deep hysteresis features are enabled by default for all clients with the feature, and the default values are considered sufficient for performance and power. Note that these registers may only be modified when the client is idle.

15.3.4.3 FIFO Clock Ratios

The register fields discussed in this section are:

- m_MCCIF_FIFOCTRL.m_MCCIF_RDCL_RDFAST
- m_MCCIF_FIFOCTRL.m_MCCIF_WRMC_CLLE2X
- m_MCCIF_FIFOCTRL.m_MCCIF_RDMC_RDFAST
- m_MCCIF_FIFOCTRL.m_MCCIF_WRCL_MCLE2X

The registers below optimize the synchronization timing in the memory client asynchronous FIFOs, usually gaining a cycle of latency back from the clock-domain crossing. When they can be used depend on the client and memory controller clock ratio.

15.3.5 Collecting Statistics

The purpose of the statistics collection feature is to allow empirical measurement of client and memory system bandwidth and latency so that system performance can be analyzed and tuned. It may also be useful for debugging problems in case of design or silicon errors. The MC and EMC statistics-collection code are built with the same control system. All registers named "MC_STAT_*" or "EMC_STAT_*" are related to stats collection. Please contact your NVIDIA FAE for more information about statistics collection.

MC can collect data such as the number transfers/sec in the memory system or for a given client; the percentage of total bandwidth used by a given client; the percentage of requests that are high priority, etc. It can also be used to measure latency of a single client.

Similar to the MC statistics, EMC statistics can collect bandwidth and latency information on the low-latency paths. EMC also has a special statistics collector dedicated to collecting total memory-bandwidth information for the dynamic power-control software driver.

15.3.6 Miscellaneous MC Registers

15.3.6.1 Module Hot Reset Sequence

Module reset involves both CAR and MC registers if there are ongoing/outstanding memory transactions. A proper module reset sequence is as follows:

- Clear <MODULE>_ENABLE to block its memory requests
- Poll the <MODULE>_OUTREQCNT till zero
- Clear module bit in CLK_RST_CONTROLLER_RST_DEVICES register to reset module
- Clear <MODULE>_HOTRESETN to clear the blocked requests seating before arbitration
- Set <MODULE>_HOTRESETN to release the hot reset
- Set module bit in CLK_RST_CONTROLLER_RST_DEVICES register to release module reset
- Set <MODULE>_ENABLE in MC_CLIENT_CTRL register to allow new requests to proceed to arbitration

15.3.6.2 MC_CLKEN_OVERRIDE and EMC_CLKEN_OVERRIDE

These should be left as CLK_GATED for normal operation. If set to CLK_ALWAYS_ON these will override the clock-gate so that the second-level clock is always running when the first-level clock is running which increases power consumption.

15.3.6.3 MC_AXI_DECERR_OVR

Setting these bits to ENABLE will cause all AXI responses from the MC to return OK status even if the MC detects a decode error.

15.3.6.4 MC_LOWLATENCY_CONFIG

This register contains a number of configuration bits to disable the low-latency bypass path for the CPU interfaces. Disabling this path would force CPU traffic into the normal arbitration, which adds considerable latency. It also contains the LL_DRAM_INTERLEAVE field that modifies low-latency data ordering expectations to be compatible with external SDRAMs that do not support interleave mode.

15.3.6.5 MC_EMEM_ADR_CFG

This register may only be modified when the memory controller clock is below 250MHz.

15.3.6.6 Client Activity Monitors

The CLIENT_ACTIVITY_MONITOR_* registers provide per-client sticky bits indicate whether a client has had EMEM requests arbitrated since the time the register was written. All bits are cleared when the register is written. This is intended as a debugging aid.

15.3.6.7 Interrupts

Both the MC and EMC have defined interrupt events. These interrupts may be individually routed to the interrupt controller by setting the INTMASK bit corresponding to the desired interrupt. If an interrupt event occurs, irrespective of the INTMASK

status, the interrupt bit in the INTSTATUS register will get set. The INTSTATUS register bits should be cleared when the interrupt event is handled by the software by writing 1 to the associated bit.

15.3.7 Miscellaneous EMC Registers

15.3.7.1 Power-Saving Options

The register fields discussed in this section are:

- EMC_SELF_REF
- EMC_DPD
- EMC_CFG.DRAM_CLKSTOP
- EMC_CFG.DRAM_ACPD
- EMC_CFG. DRAM_CLKSTOP_PDSR_ONLY
- EMC_CFG_DLL.CFG_USE_SINGLE_DLL

The EMC contains various options to explicitly put external SDRAM into various power-down states, as well as features that will dynamically attempt to lower power consumption if the EMC detects it is in an idle state.

EMC_SELF_REF and EMC_DPD can be used to issue the named power-saving command states to SDRAM. SELF_REF is particularly useful for low-power sleep states.

DRAM_CLKSTOP and DRAM_ACPD are both controls to enable dynamic power-saving modes in EMC; both these features are highly recommended. ACPD enables EMC to opportunistically put the SDRAM in either active power down or pre-charge power down states by pulling CKE low when it detects an IDLE command stream.

CLKSTOP enables the EMC to stop driving the external SDRAM clock when there are no operations ongoing. CLKSTOP should be supported by most low-power SDRAM devices; however it will probably not be supported by mainstream SDRAM devices that have an on-chip PLL; consult the vendor device documentation to ensure CLKSTOP is supported on the device. CLKSTOP is more robust than ACPD, and can be enabled/disabled without a special programming sequence. Certain devices may not be able to stop the clock unless the device is in power-down or self-refresh, in that case the DRAM_CLKSTOP_PDSR_ONLY bit should be enabled.

CFG_USE_SINGLE_DLL: shuts down one of the digital DLL's to conserve power. In Tegra 2 devices, the DDR2 data pads have been split into 2 groups located in different corners of the die. This creates a significant possibility for OCV (thermal/process/voltage variation) between the 2 groups, creating differences in the pads' delay cells that would impact DDR performance. Having localized DLL's eliminates this risk, but increases DDR2 power

15.3.7.2 Pad Auto-calibration

Tegra 2 devices have auto-calibration logic that utilizes a special compensation pad. The auto-calibration logic calculates the optimum setting for pull-up/pull-down settings for the pad output. EMC_AUTO_CAL_CONFIG and EMC_AUTO_CAL_INTERVAL control the calibration logic, while EMC_AUTO_CAL_STATUS provides access to the calculated pull-up/pull-down values sent to the pads. Calibration takes place as follows:

1. Calibration controlled by a state machine in core. Pull-up and pull-down are calibrated in sequence. When calibration starts, state machine enables comp pad, observes ZI, and steps up DRV codes from 0 to 31, one at a time, until ZI becomes 1(0) when calibrating pull up(down) and locks the code. If correct ZI value is not detected before code reaches 31, code stays at 31. After pull-up is calibrated, process is repeated for pull-down.
2. Final codes are sent to each memory pad, and may be adjusted by offset or overridden
3. Each step (AUTO_CAL_STEP) of calibration should be set to ~1us.

Recommended calibration intervals are: once at boot up, then every 10ms, 1s.

15.3.7.3 LPDDR2 DRAM Output Driver Periodic Calibration

Tegra 2 devices provide an automated method to generate periodic MRW commands to each DRAM device (per chip-select). This is specifically intended to send ZQ calibration commands to maintaining output driver accuracy of +/- 15% (compensating for voltage/temp drift). Without this periodic recalibration, the spec can only guarantee accuracy of +/- 30% (across PVT). How often these should be run depends on device thermal/voltage sensitivity and predicted thermal/voltage drift rates in system.

ZQCorrection / (TSens * Tdrift + Vsens * Vdrift) = ZQperiod

ZQperiod is specified in terms of EMC_REFRESH, w/ a calibration command launched every EMC_REFRESH * EMC_ZCAL_REF_CNT cycles. The command to be sent is specified via EMC_ZCAL_MRW_CMD, and the post-command bus-inactivity enforced via EMC_ZCAL_WAIT_CNT.

15.3.7.4 DDR2 Terminations

There are 2 sets of termination controls, ODT (control of Tegra 2 output pin ODT) turning on/off terminations in DRAM devices and CTT, Center-Tapped Terminations on Tegra 2 devices. ODT terminations are enabled/disabled via EMC_ODT_WRITE/READ. Turning on ODT enables higher DDR2 performance, but will dissipate more power.

CTT is enabled via FBIO_CFG5.CTT_TERMINATION, enabling pull-up & pull-down on DQ/DQS lines during reads. Again this allows for increased performance at the expense of power.

15.4 Memory Controller Registers

15.4.1 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 frequency-dependant 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.

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 warmboot (aka "wake from LP0") and coldboot (aka "power up") sequences. Suggested actions for the BootROM/Bootloader are noted in comments labeled "Boot requirements: ..."

15.4.1.1 EMC_INTSTATUS_0

Clear on 1-write. Init value is clear.

Interrupt Status Register

Offset: 000h | Read/Write: R/W | Reset: 0b000

Bit	Reset	Description
5	CLEAR	MRR_DIVLD_INT: LPDDR2 MRR data is available to be read. 0 = CLEAR 1 = SET

Bit	Reset	Description
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

15.4.1.2 EMC_INTMASK_0

Init value is masked.

Interrupt Mask Register

Offset: 004h | Read/Write: R/W | Reset: 0b000

Bit	Reset	Description
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

15.4.1.3 EMC_DBG_0

The DBG register is used to reconfigure the EMC during debug or chip testing.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

Debug Register

Offset: 008h | Read/Write: R/W | Reset: 0b1xxxxxxxxxxxxx10xxx00x000

Bit	Reset	Description
24	ENABLED	CFG_PRIORITY: determines the priority of cfg accesses to the DRAM. Setting this register to ENABLED gives DRAM config cycles (refresh, mrs, emrs, etc.) higher priority over real time requestors. The DISABLED setting gives the real time requestors higher priority than DRAM config cycles. Do not program to DISABLED unless for debugging. 0 = DISABLED 1 = ENABLED
10	ENABLED	AP_REQ_BUSY_CTRL: determines whether the busy signal from the auto-precharge cancellation (APC) fifo is allowed to stall requests to the EMC. 0 = DISABLED 1 = ENABLED

Bit	Reset	Description
9	MANAGED	READ_DQM_CTRL: controls whether the dqm signals during reads are managed for power (not relevant for DDR). If set to MANAGED, EMC only turns them on when necessary. If set to ALWAYS_ON, the dqm signals are enabled during non-write operation. 0 = MANAGED 1 = ALWAYS_ON
5	DISABLED	PERIODIC_QRST: specifies whether or not to periodic reset the FBIO read-data fifo during normal operation. The periodic resets can be used for graceful recovery from an intermittent failure condition; only the initial reset is absolutely required. 0 = DISABLED 1 = ENABLED
4	MRS_2	MRS_WAIT: should be set to MRS_256 when a non-mobile DRAM is used because they require a 200 cycle delay between the DLL reset and any read commands. 0 = MRS_2 1 = MRS_256
2	DISABLED	FORCE_UPDATE: causes the active state to get updated with the assembly state immediately upon writing the TIMING_CONTROL register. 0 = DISABLED 1 = ENABLED
1	ASSEMBLY	WRITE_MUX: controls whether writes to the configuration registers are done from the assembly or active state. 0 = ASSEMBLY 1 = ACTIVE
0	ACTIVE	READ_MUX: controls whether reads to the configuration registers are done from the assembly or active state. 0 = ACTIVE 1 = ASSEMBLY

15.4.1.4 EMC_CFG_0

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 BootROM during coldboot
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

Configuration Register

Offset: 00ch | Read/Write: R/W | Reset: 0b000xxx11xxxxxxx011111111xxxxxxx0

Bit	Reset	Description
31	DISABLED	DRAM_CLKSTOP: allows the DRAM controller to turn off the clock to the DRAM when it is safe to do so (no operations are ongoing, and tRFC, tMRS, tRP, etc. have all been satisfied 0 = DISABLE 1 = ENABLE
30	DISABLED	DRAM_CLKSTOP_PDSR_ONLY: clockstop (if enabled) only allowed to happen if CKE=0 (for all CKE bits associated w/ clock) 0 = DISABLED 1 = ENABLED

Bit	Reset	Description
29	NO_POWERDOWN	DRAM_ACPD: allows the DRAM controller to perform opportunistic active powerdown control using the CKE pin on the DRAM. The behavior of the powerdown control logic is controlled by the PDEX2* and *2PDEN registers. The value of DRAM_ACPD should only be changed when CKE is low, e.g., during software-controlled self-refresh or before DRAM initialization. If enabling ACPD, you should ALWAYS enable DRAM_CLKSTOP_PDSR_ONLY. Not doing so will result in sub-optimal power-down & clockstop performance. The powerdown conditions are met within a couple of cycles after the clock has stopped, so the clock must be restarted & minimum clock timings met before powerdown can be issued and clock restopped. 0 = NO_POWERDOWN 1 = ACTIVE_POWERDOWN
25	ENABLE	RESERVED: should always be set to ENABLE 0 = DISABLE 1 = ENABLE
24	ENABLE	RESERVED: should always be set to ENABLE 0 = DISABLE 1 = ENABLE
16	DISABLE	RESERVED: should always be set to DISABLE 0 = DISABLE 1 = ENABLE
15:8	0xff	PRE_IDLE_CYCLES: cycles after which an idle bank may be closed. Note that 0 is an illegal setting for PRE_IDLE_CYCLES
0	DISABLED	PRE_IDLE_EN: preemptively closes all of the banks after the EMC has been idle for PRE_IDLE_CYCLES cycles and there are banks open. PRE_IDLE_EN can be enabled if violating tRAS max is an issue. 0 = DISABLE 1 = ENABLE

15.4.1.5 EMC_ADR_CFG_0

This is used to specify the DRAM parameters. EMC will use these parameters to generate the device, row, bank, column values to the SDRAM. The ADR_CFG shadows the address configuration being used by the MC's EMEM_ADR_CFG register. It must be programmed to the same value as the EMEM_ADR_CFG register.

Boot requirements:

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

External Memory Address Configuration Register

Offset: 010h | Read/Write: R/W | Reset: 0b00xxxxx100xxxxxx10xxxxx010

Bit	Reset	Description
25:24	N1	EMEM_NUMDEV: : the number of attached devices. If more than one device is attached, the DEVSIZE, COLWIDTH, and BANKWIDTH configurations for the second device will be defined by the fields in ADR_CFG_1, while the fields in ADR_CFG will only apply to the first device 0 = N1 (Number of Devices equals 1) 1 = N2 (Number of Devices equals 2)

Bit	Reset	Description
18:16	D64MB	EMEM_DEVSIZE: size of the attached SDRAM device used to generate width of row address. 0 = D4 MB 1 = D8 MB 2 = D16 MB 3 = D32 MB 4 = D64 MB 5 = D128 MB 6 = D256 MB 7 = D512 MB 8 = D1024MB 8 = D1GB
9:8	W2	EMEM_BANKWIDTH: width of bank address of the attached SDRAM device. 1 = device internal bank equals 2 2 = device internal bank equals 4 3 = device internal bank equals 8
2:0	W9	EMEM_COLWIDTH: width of column address of the attached SDRAM device. 0 = column width equals 7 1 = column width equals 8 2 = column width equals 9 3 = column width equals 10 4 = column width equals 11

15.4.1.6 EMC_ADR_CFG_1_0

This register allows for two devices with different colwidth/bankwidth to be used without creating holes in the system-level address map.

If ADR_CFG.EMEM_NUMDEV == N2, then this register must also be programmed in addition to ADR_CFG. If the two devices are the same colwidth/bankwidth/devsize, then this can be programmed exactly the same as ADR_CFG. Otherwise, the smaller of the two devices (devsize) should be placed as Device[1] and this register should be programmed to reflect the smaller device's bankwidth, colwidth, and devsize.

Note that due to system-level address map restrictions, more DRAM bytes may be populated in EMC than are addressable in the system.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

External Memory Address Config Register, Device [1]

Offset: 014h | Read/Write: R/W | Reset: 0b0100xxxxxx10xxxxx010

Bit	Reset	Description
19:16	D64MB	EMEM1_DEVSIZE: size of the attached SDRAM device used to generate width of row address. 0 = D4 MB 1 = D8 MB 2 = D16 MB 3 = D32 MB 4 = D64 MB 5 = D128 MB 6 = D256 MB 7 = D512 MB 8 = D1024MB 8 = D1GB
9:8	W2	EMEM1_BANKWIDTH: width of bank address of the attached SDRAM device. 2 = device internal bank equals 4 3 = device internal bank equals 8
2:0	W9	EMEM1_COLWIDTH: width of column address of the attached SDRAM device. 0 = column width equals 7 1 = column width equals 8 2 = column width equals 9 3 = column width equals 10 4 = column width equals 11

15.4.1.7 EMC_REFCTRL_0

The REFCTRL register allows SW to enable or disable the refresh controller. REF_VALID should be enabled after the initialization sequence is completed.

Boot requirements

- If per-device DPD is used, the DEVICE_REFRESH_DISABLE field should be parameterized in the BCT and written by the BootROM during coldboot.
- If per-device DPD is used, the DEVICE_REFRESH_DISABLE field should be parameterized the scratch registers registers and restored by the BootROM during warmboot.
- REF_VALID field should always be set to ENABLED for normal use, no need to parameterize in BCT/scratch.

Refresh Control Register

Offset: 020h | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

Bit	Reset	Description
31	DISABLE	REF_VALID: enable refresh controller. 0 = DISABLE 1 = ENABLE
1:0	0X0	DEVICE_REFRESH_DISABLE: disables refresh to individual attached device (1 bit per dram chip-select).

15.4.1.8 EMC_PIN_0

The PIN register allows SW to control the state of the selected external DRAM pins.

Boot requirements

- Both fields in this register should be set to NORMAL during coldboot.
- Both fields in this register should be set to NORMAL during warmboot.

Controls state of selected DRAM pins

Offset: 024h | Read/Write: R/W | Reset: 0b0xxx0

Bit	Reset	Description
4	NORMAL	PIN_DQM: is used to always mask DRAM writes. This pin should only be used for initialization. Certain DRAM vendors (e.g., Samsung), 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

15.4.1.9 EMC_TIMING_CONTROL_0

The TIMING_CONTROL register is used by SW to trigger parameter updates for timing parameter registers, rdqs/quse delay controls, and some DLL controls. Writing the TIMING_UPDATE field updates the active state of these registers with the programmed assembly state. The active state is updated during a safe interval determined by the EMC. If CLKCHANGE_REQ_ENABLE is enabled, the active value will automatically be updated on completion of the clock change.

Boot requirements: Writing this register with 0x1 will trigger a timing update event, which should be used in both warmboot and coldboot sequences. .

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

Bit	Reset	Description
0	_NONE_	TIMING_UPDATE

15.4.1.10 EMC_RC_0

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

DRAM Timing Parameter

Offset: 02ch | Read/Write: R/W | Reset: 0b111111

Bit	Reset	Description
5:0	0x3f	RC: specifies the row cycle time. This is the minimum number of cycles between activate commands to the same bank. LPDDR2: ceil((tRASmin+tRPpb)/tCK)

Bit	Reset	Description
		DDR2: ceil(tRC/tCK)

15.4.1.11 EMC_RFC_0

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

DRAM Timing Parameter

Offset: 030h | Read/Write: R/W | Reset: 0b000111111

Bit	Reset	Description
8:0	0x3f	RFC: 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. LPDDR2: ceil(tRFCab/tCK) DDR2 : ceil(tRFC/tCK)

15.4.1.12 EMC_RAS_0

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

DRAM Timing Parameter

Offset: 034h | Read/Write: R/W | Reset: 0b111111

Bit	Reset	Description
5:0	0x3f	RAS: specifies the row active time. This is the minimum number of cycles between an activate command and a precharge command to the same bank. LPDDR2: ceil(tRASmin/tCK); min = 3 DDR2 : ceil(tRASmin/tCK)

15.4.1.13 EMC_RP_0

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

DRAM Timing Parameter

Offset: 038h | Read/Write: R/W | Reset: 0b111111

Bit	Reset	Description
5:0	0x3f	RP: specifies the row precharge time. This is the minimum number of cycles between a precharge command and an activate command to the same bank.

Bit	Reset	Description
		LPDDR2: ceil(tRPpb/tCK); min = 3 DDR2 : ceil(tRP/tCK)

R2something and W2something registers are independent of burst length, and are relative to the last virtual CAS_ of a command.

Start counting from the last data transfer as related to where CAS_ would be if BL = 4.

15.4.1.14 EMC_R2W_0

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

DRAM Timing Parameter

Offset: 03ch | Read/Write: R/W | Reset: 0b11111

Bit	Reset	Description
4:0	0x1f	R2W: specifies the minimum number of cycles from any read command to any write command, irrespective of bank. This parameter guarantees the read->write turn-around time on the bus. Largest programming value is 29 LPDDR2: RL + ceil(tDQSCKmax/tCK) + 2 - WL DDR2 w/o ODT: 4 DDR2 w/ ODT: 5

15.4.1.15 EMC_W2R_0

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

DRAM Timing Parameter

Offset: 040h | Read/Write: R/W | Reset: 0b11111

Bit	Reset	Description
4:0	0x1f	W2R: specifies the minimum number of cycles from a write command to a read command, irrespective of bank. Largest programming value is 29 LPDDR2/DDR2: 2 + max(2, ceil(tWTR/tCK))

15.4.1.16 EMC_R2P_0

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

DRAM Timing Parameter

Offset: 044h | Read/Write: R/W | Reset: 0b11111

Bit	Reset	Description
4:0	0x1f	R2P: specifies the minimum number of cycles from a read command to a precharge command for the same bank. LPDDR2: ceil(tRTP/tCK); min = 2; add 1 if LPDDR2-S2 DDR2: ceil(tRTP/tCK)

15.4.1.17 EMC_W2P_0

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

DRAM Timing Parameter

Offset: 048h | Read/Write: R/W | Reset: 0b11111

Bit	Reset	Description
4:0	0x1f	W2P: specifies the minimum number of cycles from a write command to a precharge command for the same bank. LPDDR2: WL + 2 + max(3, ceil(tWR/tCK)) DDR2: WL + 1 + max(2, ceil(tWR/tCK))

15.4.1.18 EMC_RD_RCD_0

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

DRAM Timing Parameter

Offset: 04ch | Read/Write: R/W | Reset: 0b011111

Bit	Reset	Description
5:0	0x1f	RD_RCD: 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. LPDDR2: ceil(tRCD/tCK); min = 3 DDR2: ceil(tRCD/tCK)

15.4.1.19 EMC_WR_RCD_0

DRAM Timing Parameter

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

Offset: 050h | Read/Write: R/W | Reset: 0b011111

Bit	Reset	Description

Bit	Reset	Description
5:0	0x1f	WR_RCD: minimum number of cycles between an activate command and a write command to the same bank. LPDDR2: ceil(tRCD/tCK); min = 3 DDR2: ceil(tRCD/tCK)

15.4.1.20 EMC_RRD_0

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

DRAM Timing Parameter

Offset: 054h | Read/Write: R/W | Reset: 0b1111

Bit	Reset	Description
3:0	0xf	RRD: specifies the Bank X Act to Bank Y Act command delay. LPDDR2: ceil(tRRD/tCK); min = 2 DDR2: ceil(tRRD/tCK)

15.4.1.21 EMC_REXT_0

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

DRAM Timing Parameter

Offset: 058h | Read/Write: R/W | Reset: 0b0001

Bit	Reset	Description
3:0	0x1	REXT: specifies the read to read delay for reads when multiple physical devices are present. LPDDR2: if (tCK >= 6) REXT = 2 else if (tCK >= 2) REXT = 3 else REXT = 4 DDR2: 1

15.4.1.22 EMC_WDV_0

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

DRAM Timing Parameter

Offset: 05ch | Read/Write: R/W | Reset: 0b0000

Bit	Reset	Description
3:0	0x0	WDV: the number of cycles to post (delay) write data from being asserted to the rams. 15 = MAX LPDDR2: WL DDR2 : WL - 1

15.4.1.23 EMC_QUSE_0

The QRST, QUSE, RDV registers specify the delays from read to internal timing signals. These fields should be set as follows:

Because DRAM uses a tristating clock (the DQS) a method is needed to deal with the ambiguity of when DQS is tristate. Only one such method is supported: QUSE.

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

DRAM Timing Parameter

Offset: 060h | Read/Write: R/W | Reset: 0b0010

Bit	Reset	Description
3:0	0x2	QUSE: tells the chip when to look for read return data. LPDDR2/DDR2: obtained from characterization

15.4.1.24 EMC_QRST_0

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

DRAM Timing Parameter

Offset: 064h | Read/Write: R/W | Reset: 0b0001

Bit	Reset	Description
3:0	0x1	QRST: time from expiration of QSAFE until reset is issued LPDDR2: RL - 2 DDR2: CL - 2

15.4.1.25 EMC_QSAFE_0

When PERIODIC_QRST is enabled, the QSAFE parameter is intended to guarantee that the QRSTs will not interfere with pending reads (e.g. the queue is empty). This field must be set to at least (RDV - QRST).

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

DRAM Timing Parameter

Offset: 068h | Read/Write: R/W | Reset: 0b0111

Bit	Reset	Description
3:0	0x7	QSAFE: time from a read command to when it is safe to issue a QRST (delayed by the QRST parameter). LPDDR2/DDR2: QSAFE >= RDV - QRST

15.4.1.26 EMC_RDV_0

RDV is the read latency register. This register value is not negotiable, it will work with the right value and will not with the wrong value. It is sequential timing, not combinational (i.e., maybe the silicon is fast enough type) timing.

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

DRAM Timing Parameter

Offset: 06ch | Read/Write: R/W | Reset: 0b01000

Bit	Reset	Description
4:0	0x8	RDV: time from read command to latching the read data from the pad macros. 15 = MAX LPDDR2: RL + 5 + ceil(tDQSCKmax/tCK) DDR2: CL + 7 for DDR2

15.4.1.27 EMC_REFRESH_0

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

DRAM Timing Parameter

The whole register value is recommended to be about 40 less than floor(min(tREFI, tRASmax)/tCK), but no less than 80% of that value.

Offset: 070h | Read/Write: R/W | Reset: 0bxxxxxxxxxxx11111

Bit	Reset	Description
15:5	X	REFRESH: specifies the interval between refresh requests (MSB portion)
4:0	MAX	REFRESH_LO: specifies the interval between refresh requests (LSB portion) 31 = MAX

15.4.1.28 EMC_BURST_REFRESH_NUM_0

BURST_REFRESH_NUM is used to specify the refresh burst count. The refresh controller will wait until BURST_REFRESH_NUM refreshes have been scheduled, and then issue them all at once. This can result in a performance

improvement in many cases. This may increase worst-case read latency due to burst refresh, so it is recommended to be programmed to BR1 for cases where latency-sensitive modules are active.

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.

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

DRAM Timing Parameter

Offset: 074h | Read/Write: R/W | Reset: 0b0000

Bit	Reset	Description
3:0	BR1	BURST_REFRESH_NUM: specify the refresh burst count. 0 = Burst Refresh count equals 1 1 = Burst Refresh count equals 2 2 = Burst Refresh count equals 4 3 = Burst Refresh count equals 8 4 = Burst Refresh count equals 16 5 = Burst Refresh count equals 32 6 = Burst Refresh count equals 64 7 = Burst Refresh count equals 128 8 = Burst Refresh count equals 256 9 = Burst Refresh count equals 512 9 = MAX

15.4.1.29 EMC_PDEX2WR_0

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

DRAM Timing Parameter

Offset: 078h | Read/Write: R/W | Reset: 0b1110

Bit	Reset	Description
3:0	0xe	PDEX2WR: specify the timing delay from exit of powerdown mode to a write command. Largest allowed value is 14 LPDDR2/DDR2: ceil(tXP/tCLK)+1

15.4.1.30 EMC_PDEX2RD_0

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.

- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

DRAM Timing Parameter

Offset: 07ch | Read/Write: R/W | Reset: 0b1110

Bit	Reset	Description
3:0	0xe	PDEX2RD: specify the timing delay from exit of powerdown mode to a read command. Largest allowed value is 14 LPDDR2/DDR2: ceil(tXP/tCLK)+1

15.4.1.31 EMC_PCHG2PDEN_0

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

DRAM Timing Parameter

Offset: 080h | Read/Write: R/W | Reset: 0b01111

Bit	Reset	Description
4:0	0xf	PCHG2PDEN: specify the timing delay from a precharge command to powerdown entry. LPDDR2/DDR2: ceil(tRP/tCK)

15.4.1.32 EMC_ACT2PDEN_0

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

DRAM Timing Parameter

Offset: 084h | Read/Write: R/W | Reset: 0b01111

Bit	Reset	Description
4:0	0xf	ACT2PDEN: specify the timing delay from an activate, mrs or emrs command to powerdown entry. LPDDR2: max(8, ceil(tRCD/tCK)) DDR2: ceil(tRCD/tCLK)

15.4.1.33 EMC_AR2PDEN_0

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

DRAM Timing Parameter

Offset: 088h | Read/Write: R/W | Reset: 0b11111

Bit	Reset	Description
4:0	0x1f	AR2PDEN: specify the timing delay from an autorefresh command to powerdown entry. LPDDR2/DDR2: 1

15.4.1.34 EMC_RW2PDEN_0

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

DRAM Timing Parameter

Offset: 08ch | Read/Write: R/W | Reset: 0b001111

Bit	Reset	Description
5:0	0xf	RW2PDEN: specify the timing delay from a read/write command to powerdown entry. Auto-precharge timing must be taken into account when programming this field. LPDDR2: max(RL+ceil((tDQSCKmax+tRP)/tCK)+1, WL+1+ceil((tWR+tRP)/tCK)) DDR2: max(RL+ceil((tDQSCKmax+tRP)/tCK)+1, WL+ceil((tWR+tRP)/tCK))

15.4.1.35 EMC_TXSR_0

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

DRAM Timing Parameter

Offset: 090h | Read/Write: R/W | Reset: 0b011111111111

Bit	Reset	Description
11:0	0x7ff	TXSR: cycles between self-refresh exit & first DRAM command Largest allowed value is 0xffe LPDDR2: ceil(tXSR/tCK); min = 2 DDR2: ceil(tXSR/tCK)

15.4.1.36 EMC_TCKE_0

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

DRAM Timing Parameter

Offset: 094h | Read/Write: R/W | Reset: 0b1110

Bit	Reset	Description
3:0	0xe	TCKE: specify minimum CKE pulse width. LPDDR2: max(tCKE, ceil(tXP/tCKmin)): tCKmin is the tCK of the max frequency of the DRAM parts used DDR2: tCKE

15.4.1.37 EMC_TFAW_0

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

DRAM Timing Parameter

Offset: 098h | Read/Write: R/W | Reset: 0b000000

Bit	Reset	Description
5:0	0x0	TFAW: 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 withing the rolling window. LPDDR2: ceil(tFAW/tCK); min = 8 DDR2: ceil(tFAW/tCK)

15.4.1.38 EMC_TRPAB_0

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

DRAM Timing Parameter

Offset: 09ch | Read/Write: R/W | Reset: 0b000000

Bit	Reset	Description
5:0	0x0	TRPAB: specify precharge-all tRP allowance for 8-bank devices. Setting this field to 0 will cause EMC to use TRP.TRP for precharge-all. LPDDR2: ceil((tRPpb+3)/tCK); min = 3 DDR2: ceil((tRP + tCK)/tCK)

15.4.1.39 EMC_TCLKSTABLE_0

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

DRAM Timing Parameter

Offset: 0a0h | Read/Write: R/W | Reset: 0b0000

Bit	Reset	Description
3:0	0x0	TCLKSTABLE: specify minimum number of cycles of a stable clock period prior to exiting powerdown or self-refresh modes. LPDDR2: max(5, ceil(tRFC]/tCKmin/4) - 2): tCKmin is the tCK of the max frequency of the DRAM parts used DDR2: 2

15.4.1.40 EMC_TCLKSTOP_0

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

DRAM Timing Parameter

Offset: 0a4h | Read/Write: R/W | Reset: 0b0010

Bit	Reset	Description
3:0	0x2	TCLKSTOP: delay from last command to stopping the external clock to DRAM devices. LPDDR2/DDR2: 2

15.4.1.41 EMC_TREFBW_0

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

DRAM Timing Parameter

Offset: 0a8h | Read/Write: R/W | Reset: 0b00000000000000

Bit	Reset	Description
13:0	0x0	TREFBW: 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. LPDDR2: ceil(32*tRFCab/tCK) DDR2: 0

15.4.1.42 EMC_QUSE_EXTRA_0

QUSE_EXTRA is combined with quse to add extra cycles to data return window. This is for use with dqs pulldowns in lpddr2 where start of read preamble could vary by more than 1 cycle

Example: To extend end of QUSE window by 1 cycle, set QUSE_EXTRA = QUSE + 1. To start quse window earlier by 1 cycle, set QUSE_EXTRA = QUSE - 1. Disabled if QUSE_EXTRA = 0.

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

Offset: 0ach | Read/Write: R/W | Reset: 0b0000

Bit	Reset	Description
3:0	0x0	QUSE_EXTRA

15.4.1.43 EMC_ODT_WRITE_0

For DDR2, ODT may be enabled during writes and disabled during reads via control of ODT pin.

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

Offset: 0b0h | Read/Write: R/W | Reset: 0b00xxxxxxxxxxxxxxxxxxxxxxxxxxxx000

Bit	Reset	Description
31	0x0	ENABLE_ODT_DURING_WRITE: enables ODT to be turned on prior to issuing write to DRAM. If ENABLE_ODT_DURING_WRITE = 1 and DISABLE_ODT_DURING_READ = 0, ODT will always be enabled after 1st write.
30	0x0	ODT_B4_WRITE: 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.
2:0	0x0	ODT_WR_DELAY: 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

15.4.1.44 EMC_ODT_READ_0

For DDR2, ODT may be enabled during writes and disabled during reads via control of ODT pin.

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

Offset: 0b4h | Read/Write: R/W | Reset: 0b00xxxxxxxxxxxxxxxxxxxxxxxxxxxx000

Bit	Reset	Description
31	0x0	DISABLE_ODT_DURING_READ: enables ODT to be turned off prior to issuing read to DRAM. If this field == 0, ODT state will not be changed for reads. If this field == 1, Turn off ODT prior to READ command (has no effect if ODT ENABLE_ODT_DURING_WRITE == 0, as ODT will always be disabled).
30	0x0	ODT_B4_READ: If this field == 1, ODT is turned off ODT_RD_DELAY cycles prior to dram READ command. If this field == 0, ODT is turned off ODT_RD_DELAY cycles after dram READ command. Set ODT_B4_READ to 1 if (RL - ceiling(tAOFD) - 2) < 0.
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 <=

Bit	Reset	Description
		ODT_RD_DELAY <= 1 if ODT_B4_READ=1

5 spaces reserved for timing register expansion

15.4.1.45 EMC_MRS_0

The MRS register allows SW to issue an MRS command.

BA0, BA1 are used to address MRS or EMRS registers in DRAM. Although this register can also program EMRS, use the EMRS register so that the HW registers can shadow what is in the DRAM.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.

Command trigger: MRS

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

Bit	Reset	Description
31:30	X	MRS_DEV_SELECTN: active low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for only dev1.
21:20	X	MRS_BA: Set to 0x0 for MRS.
13:0	X	MRS_ADR: mode-register data to be written.

15.4.1.46 EMC_EMRS_0

The EMRS register allows SW to issue an EMRS command.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.

Command trigger: EMRS

Offset: 0d0h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:30	X	EMRS_DEV_SELECTN: active low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for only dev1.
21:20	X	EMRS_BA: Set to 0x1 for EMRS (and where applicable, 0x2 for EMRS2, and 0x3 for EMRS3).
13:0	X	EMRS_ADR: mode-register data to be written.

15.4.1.47 EMC_REF_0

The REF register allows SW to issue refresh commands. This is done to ensure proper DRAM initialization.

Boot requirements

- This register triggers a refresh command. REF_CMD should be written with 0x1 during coldboot to trigger refresh commands during DRAM initialization.

Command trigger: Refresh

Offset: 0d4h | Read/Write: R/W | Reset: 0b00000000xxxxxxx0

Bit	Reset	Description
15:8	0x0	REF_NUM: perform (REF_NUM + 1) refresh cycles.
0	0x0	REF_CMD: causes the hardware to perform a REFRESH to all DRAM banks.

15.4.1.48 EMC_PRE_0

The PRE register allows SW to issue a precharge all command. This command may be used to ensure proper DRAM initialization.

Boot requirements

- This register triggers a precharge-all command. PRE_CMD should be written with 0x1 during coldboot 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 BootROM during coldboot.

Command trigger: Precharge-All

Offset: 0d8h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0

Bit	Reset	Description
31:30	X	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.

15.4.1.49 EMC_NOP_0

The NOP register allows SW to issue an explicit nop command. This command may be used to ensure proper DRAM initialization.

Boot requirements

- This register triggers a no-operation command. NOP_CMD should be written with 0x1 during coldboot to trigger no-op commands during DRAM initialization.

Command trigger: NOP

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

Bit	Reset	Description
0	0x0	NOP_CMD: causes the hardware to perform a NOP to all DRAM banks.

15.4.1.50 EMC_SELF_REF_0

The SELF_REF register allows SW to issue self-refresh commands.

Command trigger: SELF REFRESH

Offset: 0e0h | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxxxxxxxxxxxxxxxxxx0

Bit	Reset	Description
31:30	0x3	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

Bit	Reset	Description
		will override the PIN:CKE setting. The DRAM will ignore all accesses until CMD:DISABLED. 0 = DISABLED 1 = ENABLED

15.4.1.51 EMC_DPD_0

The DPD register allows SW to issue a deep power down command.

Command trigger: Deep Power Down

Offset: 0e4h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0

Bit	Reset	Description
31:30	X	DPD_DEV_SELECTN: active low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for only dev1.
0	DISABLED	DPD_CMD: causes the hardware to issue the deep power down command (Burst Terminate w/ 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

15.4.1.52 EMC_MRW_0

Mode Register Write: LPDDR2-only version of MRS/EMRS

Boot requirements

This register triggers a mode register write command. Multiple mode-register writes may be required by the coldboot sequence. Such commands should be parameterized in the BCT and written by the BootROM during coldboot.

Command trigger: MRW

Offset: 0e8h | Read/Write: WO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:30	X	MRW_DEV_SELECTN: active-low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for dev1.
23:16	X	MRW_MA: register address
7:0	X	MRW_OP: data to be written

15.4.1.53 EMC_MRR_0

Mode Register Read: LPDDR2 only

Sequence

1. read MRR until EMC_STATUS.MRR_DIVLD=0 (ok to skip if it is sure there is no pending MRR reads)

2. write this register with the desired addr (MA) and device (DEV_SELECTN), device needs to be either DEV0 or DEV1: writing to both is illegal

3. poll EMC_STATUS.MRR_DIVLD=1, or if using interrupt, wait for MRR_DIVLD_INT

4. read back MRR. The value is in MRR_DATA field.

Note: New MRR requests may be issued while there are on-going requests. e.g. to issue 3 MRR, follow these steps: 1,2,2,2,3,4,3,4,3,4. Data read back in step 4 are in the same order requested in step 2.

To make sure EMC is available for new MRR requests, poll for EMC_STATUS.MRR_FIFO_SPACE > 0 before step 2.

If using 2 x16 DRAM, MRR_DATA[15:8] can be used to store MRR from 2nd DRAM on same CS (configured via CFG_2.MRR_BYTESEL_X16)

Command trigger: MRR

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

Bit	Reset	Description
31:30	X	MRR_DEV_SELECTN: active-low chip-select, choose which device to send the command to. (enum for safety). 0 = ILLEGAL 1 = DEV1 2 = DEV0 3 = RESERVED
23:16	X	MRR_MA: register address
15:0	X	MRR_DATA: data returned

15.4.1.54 EMC_FBIO_CFG1_0

The FBIO_CFG1 register controls the width of the DQ enable window when driving write data.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

FBIO Configuration Register

Offset: 0f4h | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
16	DISABLE	CFG_DEN_EARLY: determines whether the output enable is the same width as data (DEN_EARLY=0) or 1/2 bit time wider on either end (DEN_EARLY=1). 0 = DISABLE 1 = ENABLE

15.4.1.55 EMC_FBIO_DQSIB_DLY_0

The FBIO_DQSIB_DELAY register sets the trimmers for inbound dqs delay on each of the inbound byte lanes.

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

FBIO Configuration Register

Offset: 0f8h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:24	X	CFG_DQSIB_DLY_BYTE_3

Bit	Reset	Description
23:16	X	CFG_DQSIB_DLY_BYTE_2
15:8	X	CFG_DQSIB_DLY_BYTE_1
7:0	X	CFG_DQSIB_DLY_BYTE_0

15.4.1.56 EMC_FBIO_DQSIB_DLY_MSB_0

FBIO_DQSIB_DLY_MSB contains the most-significant-bits of values in FBIO_DQSIB_DLY.

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

FBIO Configuration Register

Offset: 0fch | Read/Write: R/W | Reset: 0b00xxxxxx00xxxxxx00xxxxxx00

Bit	Reset	Description
25:24	0x0	CFG_DQSIB_DLY_MSB_BYTE_3
17:16	0x0	CFG_DQSIB_DLY_MSB_BYTE_2
9:8	0x0	CFG_DQSIB_DLY_MSB_BYTE_1
1:0	0x0	CFG_DQSIB_DLY_MSB_BYTE_0

15.4.1.57 EMC_FBIO_CFG5_0

The FBIO_CFG5 register controls the FBIO I/O cells.

The following fields are shadowed: DIFFERENTIAL_DQS, CTT_TERMINATION, DQS_PULLD.

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 BootROM during coldboot.
- This register (except for field DISABLE_CONCURRENT_AUTOPRE) should be saved in the scratch registers and restored by the BootROM during warmboot.

FBIO Configuration Register

Offset: 104h | Read/Write: R/W | Reset: 0b0000xx0xx01

Bit	Reset	Description
10	DISABLED	DISABLE_CONCURRENT_AUTOPRE: disables reads/writes to a device until the precharge command has been issued by the dram internally. 0 = DISABLED 1 = ENABLED
9	DISABLED	DQS_PULLD: enables pulldowns on dqs lines (and pullups on DQS_N if DIFFERENTIAL_DQS). 0 = DISABLED 1 = ENABLED

Bit	Reset	Description
8	DISABLED	CTT_TERMINATION: enables CTT_TERMINATION mode in pads (ddr2 support) 0 = DISABLED 1 = ENABLED
7	DISABLED	DIFFERENTIAL_DQS: enables differential signalling on dqs strobes (lpddr2/ddr2 options) 0 = DISABLED 1 = ENABLED
4	X32	DRAM_WIDTH: specifies whether the DRAM data-bus is 16-bits or 32-bits wide. 0 = X32 1 = X16
1:0	DDR1	DRAM_TYPE: specifies which DRAM protocol to use for the attached device(s). 0 = RESERVED 1 = DDR1 2 = LPDDR2 3 = DDR2

15.4.1.58 EMC_FBIO_QUSE_DLY_0

QUSE_LATE and QUSE_DLY determine how much added delay fbio should add to the QUSE path. QUSE needs to align (approximately) with the incoming DQS (DDR1) in order to qualify it, since the incoming DQS/feedback clock is not always valid.

DRAMC can position QUSE with m2clk granularity (2 bit times). For DDR1 only, QUSE_LATE provides finer granularity of 1/2 an m4clk cycle (1/2 bit time). The amount of delay we add will be primarily a function of the round trip wire delay to/from the DRAM. Other portions of the delay (driver and receiver delay) are compensated for by delay through a non-bonded QUSE pad cell.

QUSE_DLY controls trimmers in each byte lane. This also allows for variation in the propagation delay from the "common" pad macro where the QUSE logic lives out to each of the 4 byte lanes.

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

FBIO Configuration Register

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

Bit	Reset	Description
31:24	X	CFG_QUSE_DLY_BYTE_3
23:16	X	CFG_QUSE_DLY_BYTE_2
15:8	X	CFG_QUSE_DLY_BYTE_1
7:0	X	CFG_QUSE_DLY_BYTE_0

15.4.1.59 EMC_FBIO_QUSE_DLY_MSB_0

FBIO_QUSE_DLY_MSB contains the most-significant-bits of values in FBIO_QUSE_DLY

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.

- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

FBIO Configuration Register

Offset: 110h | Read/Write: R/W | Reset: 0b00xxxxxx00xxxxxx00xxxxxx00

Bit	Reset	Description
25:24	0x0	CFG_QUSE_DLY_MSB_BYTE_3
17:16	0x0	CFG_QUSE_DLY_MSB_BYTE_2
9:8	0x0	CFG_QUSE_DLY_MSB_BYTE_1
1:0	0x0	CFG_QUSE_DLY_MSB_BYTE_0

15.4.1.60 EMC_FBIO_CFG6_0

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

FBIO Configuration Register

Offset: 114h | Read/Write: R/W | Reset: 0b010

Bit	Reset	Description
2:0	0x2	CFG_QUSE_LATE

15.4.1.61 EMC_DQS_TRIMMER_RD0_0

The DQS_TRIMMER_RD registers provide a way to observe the values being used by the trimmers

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

Bit	Reset	Description
25:16	X	DQS_CURRENT_TRIM_VAL_BYTE_0
9:0	X	QUSE_CURRENT_TRIM_VAL_BYTE_0

15.4.1.62 EMC_DQS_TRIMMER_RD1_0

Offset: 124h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25:16	X	DQS_CURRENT_TRIM_VAL_BYTE_1
9:0	X	QUSE_CURRENT_TRIM_VAL_BYTE_1

15.4.1.63 EMC_DQS_TRIMMER_RD2_0

Offset: 128h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25:16	X	DQS_CURRENT_TRIM_VAL_BYTE_2
9:0	X	QUSE_CURRENT_TRIM_VAL_BYTE_2

15.4.1.64 EMC_DQS_TRIMMER_RD3_0

Offset: 12ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25:16	X	DQS_CURRENT_TRIM_VAL_BYTE_3
9:0	X	QUSE_CURRENT_TRIM_VAL_BYTE_3

15.4.1.65 EMC_LL_ARB_CONFIG_0

Configuration of arbiter between regular MC path and low-latency path.

DIE_OFF_EXP: Exponent of the die-off factor, which is used to calculate the average load of the regular data transaction path. Valid values are from 0 to 8. The die-off factor is calculated as 1 / (2^DIE_OFF_EXP).

0: Only the last transaction is used to calculated the load

8: minimal exponential die-off factor of 1/256

ALLOW_IDLE_INSERT: If enabled, arbiter would allow a low-latency request when the MC<=>EMC path is idle. If disabled, the low-latency request would wait till the insertion delay programed.

MAX_LL_GREED: Specify the maximum number of consecutive 256-bit transactions the low-latency path is allowed to perform. The amount of transactions is not only limited by this number: transactions also have to be in the same page and in the same direction as the previous one.

For streaming read and write operations, where this is the most useful, transactions will switch page every 64 bytes, due to the location of the bank bits in the address. The greed counter has a granularity of 32 bytes, so in practise, the only practically useful numbers are 1 and 2.

In addition to allowing scheduling a consecutive transaction, the greed counter also guarantees that autopc will be issued at the end of the programmed number of consecutive writes.

Potential values:

0: 1 transaction executed at a time. When the next transaction (already in FIFO) is to the same page, autopc will not be set.

1: 1 transaction executed at a time. Autopc will always be issued.

2: 2 transactions can be executed, if they go to the same page. Autopc is issued at the end of the 2nd transaction or after the first one if there is no second transaction.

3 and higher: Same as 2, but for more transactions. In practice, this will not happen since the address mapping is such that bank changes after 2 transactions.

Boot requirements

- This arbitration configuration register should be parameterized in the BCT and written by the OS during coldboot and warmboot.

LOW-LATENCY Arbiter Configuration

Offset: 144h | Read/Write: R/W | Reset: 0b000000xxxxxxxx0010xxx0xxxx0011

Bit	Reset	Description
29	0x0	LL_INSERT_DIFF_BANK_AFTER_REMOVE: set to zero to get AP15 behavior
28	0x0	LL_INSERT_DIFF_BANK_AFTER_TOGGLE: set to one to get AP15 behavior
27	0x0	LL_INSERT_DIFF_BANK_BEFORE_DISABLE: set to one to get AP15 behavior

Bit	Reset	Description
26	0x0	LL_FORCE_INSERT_WR_DISABLE: set to one to get AP15 behavior
25	0x0	LL_GREED_DIFF_BANK_AFTER_DISABLE: set to one to get AP15 behavior
24	0x0	LL_GREED_DIFF_BANK_DISABLE: set to one to get AP15 behavior
15:12	0x2	MAX_LL_GREED
8	DISABLED	ALLOW_IDLE_INSERT: 0 = DISABLED 1 = ENABLED
3:0	0x3	DIE_OFF_EXP

15.4.1.66 EMC_T_MIN_CRITICAL_HP_0

Configuration table with low-latency insertion delay based on the criticality level inside the MC. Agents inside the MC that can indicate critical status. Depending on the number of agents that are currently in critical status, a minimum value can be programmed before which a low-latency transfer is allowed.

There are 4 levels for high priority: 0, 1, 2, 3_or_more

Boot requirements

- This arbitration configuration register should be parameterized in the BCT and written by the OS during coldboot and warmboot.

LOW-LATENCY Arbiter Configuration

Offset: 148h | Read/Write: R/W | Reset: 0b00001010000010000000011000000000

Bit	Reset	Description
31:24	0xa	T_MIN_CRIT_HP_3
23:16	0x8	T_MIN_CRIT_HP_2
15:8	0x6	T_MIN_CRIT_HP_1
7:0	0x0	T_MIN_CRIT_HP_0

15.4.1.67 EMC_T_MIN_CRITICAL_TIMEOUT_0

There are 4 levels for timeout: 0, 1, 2, 3_or_more

Boot requirements

- This arbitration configuration register should be parameterized in the BCT and written by the OS during coldboot and warmboot.

LOW-LATENCY Arbiter Configuration

Offset: 14ch | Read/Write: R/W | Reset: 0b00001010000010000000011000000000

Bit	Reset	Description
31:24	0xa	T_MIN_CRIT_TIMEOUT_3
23:16	0x8	T_MIN_CRIT_TIMEOUT_2
15:8	0x6	T_MIN_CRIT_TIMEOUT_1
7:0	0x0	T_MIN_CRIT_TIMEOUT_0

15.4.1.68 EMC_T_MIN_LOAD_0

There are 4 level of (time averaged) load. For each of those levels, a different minimum low-latency hold-off value can be programmed.

Boot requirements

- This arbitration configuration register should be parameterized in the BCT and written by the OS during coldboot and warmboot.

LOW-LATENCY Arbiter Configuration

Offset: 150h | Read/Write: R/W | Reset: 0b00001000000001000000001000000000

Bit	Reset	Description
31:24	0x8	T_MIN_LOAD_3
23:16	0x4	T_MIN_LOAD_2
15:8	0x2	T_MIN_LOAD_1
7:0	0x0	T_MIN_LOAD_0

15.4.1.69 EMC_T_MAX_CRITICAL_HP_0

Maximum hold-off time for a given criticality in the MC.

Boot requirements

- This arbitration configuration register should be parameterized in the BCT and written by the OS during coldboot and warmboot.

LOW-LATENCY Arbiter Configuration

Offset: 154h | Read/Write: R/W | Reset: 0b00001011000010100000100100000001

Bit	Reset	Description
31:24	0xb	T_MAX_CRIT_HP_3
23:16	0xa	T_MAX_CRIT_HP_2
15:8	0x9	T_MAX_CRIT_HP_1
7:0	0x1	T_MAX_CRIT_HP_0

15.4.1.70 EMC_T_MAX_CRITICAL_TIMEOUT_0

Boot requirements

- This arbitration configuration register should be parameterized in the BCT and written by the OS during coldboot and warmboot.

LOW-LATENCY Arbiter Configuration

Offset: 158h | Read/Write: R/W | Reset: 0b00001011000010100000100100000001

Bit	Reset	Description
31:24	0xb	T_MAX_CRIT_TIMEOUT_3
23:16	0xa	T_MAX_CRIT_TIMEOUT_2
15:8	0x9	T_MAX_CRIT_TIMEOUT_1

Bit	Reset	Description
7:0	0x1	T_MAX_CRIT_TIMEOUT_0

15.4.1.71 EMC_T_MAX_LOAD_0

Maximum load dependent hold-off time

Boot requirements

- This arbitration configuration register should be parameterized in the BCT and written by the OS during coldboot and warmboot.

LOW-LATENCY Arbiter Configuration

Offset: 15ch | Read/Write: R/W | Reset: 0b00100000000100000000100000000100

Bit	Reset	Description
31:24	0x20	T_MAX_LOAD_3
23:16	0x10	T_MAX_LOAD_2
15:8	0x8	T_MAX_LOAD_1
7:0	0x4	T_MAX_LOAD_0

15.4.1.72 EMC_AUTO_CAL_CONFIG_0

Boot requirements

- This register (except for AUTOCAL_SLW_OVERRIDE and AUTO_CAL_START(trigger)) should be parameterized in the BCT and written by the BootROM during coldboot.
- This register (except for AUTOCAL_SLW_OVERRIDE and AUTO_CAL_START(trigger)) should be saved in the scratch registers and restored by the BootROM during warmboot.

Auto-calibration Settings for EMC Pads

Offset: 2a4h | Read/Write: R/W | Reset: 0b0000xx0010100110xxx00000xxx00000

Bit	R/W	Reset	Description
31	RO	0x0	AUTO_CAL_START: Writing a one to this bit starts the calibration state machine. This bit must be set even if the override is set in order to latch in the override value.
30	RW	0x0	AUTO_CAL_OVERRIDE: 0 (normal operation): use AUTO_CAL_PU/PD_OFFSET as an offset to the calibration tate machine setting 1 (override) : use AUTO_CAL_PU/PD_OFFSET register values directly
29	RW	DISABLED	AUTO_CAL_ENABLE: 1 (normal operation): use EMC generated pullup/dn (override or autocal) 0 (disabled): use cfg2tmc_xm2* register settings for pullup/dn 0 = DISABLED 1 = ENABLED
28	RW	0x0	AUTOCAL_SLW_OVERRIDE: 0 (Normal operation) pad DRVDN/UP_SLWR/F tied to AUTO_CAL output DRDVDN/UP_SLWR/F[3:0] = AUTO_CAL_PULLDOWN/UP[4:1] 1 (override) use CFG2TMC_*_DRVDN/UP_SLWR/F pins to control pad slew inputs
25:16	RW	0xa6	AUTO_CAL_STEP: Auto Cal calibration step interval (in emc clocks) - the default is set for 1.0us calibration step at 166MHz
12:8	RW	0x0	AUTO_CAL_PD_OFFSET: 2's complement offset for pull-down value
4:0	RW	0x0	AUTO_CAL_PU_OFFSET: 2's complement offset for pull-up value

15.4.1.73 EMC_AUTO_CAL_INTERVAL_0

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

EMC Pad Calibration Interval

Offset: 2a8h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
27:0	0x0	AUTO_CAL_INTERVAL: 0: do calibration once Otherwise, auto-calibration occurs at intervals equivalent to the programmed number of cycles.

15.4.1.74 EMC_AUTO_CAL_STATUS_0

EMC pad calibration status

Offset: 2ach | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	AUTO_CAL_ACTIVE: One when auto calibrate is active - valid only after auto calibrate sequence has completed (EMC_CAL_ACTIVE == 0)
28:24	X	AUTO_CAL_PULLDOWN_ADJ: Pulldown code sent to pads
20:16	X	AUTO_CAL_PULLUP_ADJ: Pullup code sent to pads
12:8	X	AUTO_CAL_PULLDOWN: Pulldown code generated by auto-calibration
4:0	X	AUTO_CAL_PULLUP: Pullup code generated by auto-calibration

15.4.1.75 EMC_REQ_CTRL_0

Request status/control

Offset: 2b0h | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
1	0x0	STALL_ALL_WRITES: Stall incoming write transactions
0	0x0	STALL_ALL_READS: Stall incoming read transactions (1st non-LL read will stall all transactions)

15.4.1.76 EMC_EMC_STATUS_0

DRAM_IN_POWERDOWN, DRAM_IN_SELF_REFRESH, DRAM_IN_DPD: active high signal indicating current DRAM status for each of these modes, w/ 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).

EMC State-Machine Status

Offset: 2b4h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
20	X	MRR_DIVLD: mrr data available for reading
19:16	X	MRR_FIFO_SPACE: mrr fifospace available
13:12	X	DRAM_IN_DPD: dev[n] has been put into deep powerdown state
9:8	X	DRAM_IN_SELF_REFRESH: dev[n] has been put into self-refresh (will remain until SR exit cmd).
5:4	X	DRAM_IN_POWERDOWN: dev[n] has entered powerdown state (incoming req's will awaken if not stalled)
2	X	NO_OUTSTANDING_TRANSACTIONS: All non-stalled requests have completed
1	X	EMC_LL_REQ_FIFO_EMPTY: LL Request fifo is empty
0	X	EMC_REQ_FIFO_EMPTY: Request fifo is empty

15.4.1.77 EMC_CFG_2_0

Clock change sequencing: Once the divider is reprogrammed, CAR signals to EMC that a clock change is pending. If enabled, EMC stalls incoming requests, drains outstanding requests, and, if CLKCHANGE_(PD|SR)_ENABLE is enabled, puts DRAM into powerdown (or self-refresh) before signalling to CAR that it is idle and ready for the change to happen.

CAR will then change the divider/pll reprogramming. Once complete, EMC updates its shadow registers (assuming they may have been reprogrammed for new clock setting), unstalls requests, and resumes operation with new clock settings.

Boot requirements

- This register (except for fields DRAMC_PRE_B4_ACT, MRR_BYTESEL_X16, MRR_BYTESEL) should be parameterized in the BCT and written by the BootROM during coldboot.
- 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 BootROM during warmboot.
- 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.

EMC Configuration

Offset: 2b8h | Read/Write: R/W | Reset: 0b0xxxxxxxxx00xx00xxxxx000xxxxx011

Bit	Reset	Description
31	DISABLED	DRAMC_PRE_B4_ACT: CYA bit, gives priority to activates over precharges, determining which (precharge/activate) is processed first if both are pending and unblocked. 0 = DISABLED 1 = ENABLED
21:20	0x0	MRR_BYTESEL_X16: If using 2 X16 DRAM on a single CS to form 32-bit wide data, indicates which bytelane 2nd DRAM's byte 0 is connected to.
17:16	0x0	MRR_BYTESEL: Indicates which AP bytelane is connected to DRAM byte 0 (over which MRR data is returned).
10	DISABLED	USE_ADDR_CLK: Used to select source for DRAM clock. If enabled, xm2_addr_mclk pins instead of xm2_mclk. the former is located adjacent to addr pins used in lpddr2 (for lower clk to addr skew). If disabled, xm2_addr_mclk will be disabled & xm2_mclk will output DRAM clock (required for LPDDR_POP). 0 = DISABLED 1 = ENABLED
9:8	LPDDR2	PIN_CONFIG: Remaps address/command pins for LPDDR_POP ball-out otherwise uses standard LPDDR2 pin configuration. 0 = LPDDR2

Bit	Reset	Description
		1 = LPDDR_POP 2 = RESERVED
2	DISABLED	CLKCHANGE_SR_ENABLE: Forces dram into self-refresh during CLKCHANGE. Takes precedent over CLKCHANGE_PD_ENABLE if both are set. 0 = DISABLED 1 = ENABLED
1	ENABLED	CLKCHANGE_PD_ENABLE: Forces dram into power-down during CLKCHANGE. 0 = DISABLED 1 = ENABLED
0	ENABLED	CLKCHANGE_REQ_ENABLE: allows EMC and CAR to handshake on PLL divider/source changes. 0 = DISABLED 1 = ENABLED

15.4.1.78 EMC_CFG_DIG_DLL_0

Controls for digital DLL's, which used to measure and maintain 1/4 cycle phase adjustment for RDQS strobes.

This register is shadowed.

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 BootROM during coldboot.
- 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 BootROM during warmboot.

Configure Digital DLL

Offset: 2bch | Read/Write: R/W | Reset: 0bx0000x0000000000xxxx000001010111

Bit	R/W	Reset	Description
31	RO	X	CFG_DLL_USE_OVERRIDE_UNTIL_LOCK: Writing 1 to this register causes override_val to be used in place of DLL output until DLL_LOCK is obtained. Takes effect on next shadow update.
30	RO	0x0	DLL_RESET: Writing 1 to this register will send reset pulse to DLL's on next shadow update. Must reset DLL's when changing clock frequency by factor >= 2
29:28	RW	0x0	CFG_DLL_LOCK_LIMIT: CYA in case DLL has problems locking. DLL will be treated as locked after LIMIT emcclk cycles. Counter is reset w/ DLL_RESET (from above) or w/ each periodic update (if using RUN_PERIODIC). Settings are: 00: LIMIT = 2^12 01: LIMIT = 2^15 10: LIMIT = 2^16 11: LIMIT = 2^16 + 2^17
27	RW	0x0	CFG_DLL_ALARM_DISABLE: CYA bit -- disable override of DLL logic when DLL_ALM is set (otherwise overrides DLI to 0x3FF).
25:16	RW	0x0	CFG_DLL_OVERRIDE_VAL: Value to use in place of DLI output if CFG_DLL_OVERRIDE_EN is set.
11:8	RW	0x0	CFG_DLL_UDSET: DLL Loop filter control (2^(udset+3)).
7:6	RW	RUN_TIL_LOCK	CFG_DLL_MODE: Controls how frequently DLL runs, as follows 0 = RUN_CONTINUOUS : DLL will run continuously (only disabled during reads). This option will consume the most power. 1 = RUN_TIL_LOCK : after DLL_RESET is set, DLL will run until it has locked, then be disabled 2 = RUN_PERIODIC : DLL will be re-enabled w/ each refresh to make sure LOCK is maintained

Bit	R/W	Reset	Description
			3 = RESERVED
5	RW	0x0	CFG_DLL_LOWSPEED: Enable DLL for use w/ lowspeed EMCCLK operation (<200MHz).
4	RW	ENABLED	CFG_PERBYTE_TRIMMER_OVERRIDE: Set trimmer values directly for each byte via FBIO_QUSE_DLY/FBIO_DQS_DLY & FBIO_QUSE_DLY_MSB/FBIO_DQS_DLY_MSB. 0 = DISABLED 1 = ENABLED
3	RW	DISABLED	CFG_USE_SINGLE_DLL: Turn off upper DLL & use lower dll output to drive all trimmers. 0 = DISABLED 1 = ENABLED
2	RW	ENABLED	CFG_DLL_OVERRIDE_EN: Override DLL's DLI output w/ OVERRIDE_VAL (still uses mult/offset). 0 = DISABLED 1 = ENABLED
1	RW	ENABLED	CFG_DLI_TRIMMER_EN: Enable DL trimmer cells (embedded in pads). 0 = DISABLED 1 = ENABLED
0	RW	ENABLED	CFG_DLL_EN: Enable digital DLL's. 0 = DISABLED 1 = ENABLED

15.4.1.79 EMC_DLL_XFORM_DQS_0

XFORM_DQS_MULT and XFORM_DQS_OFFS are a multiplier and offset, respectively that can be used to modify the dll output.

Final products may be read out in DQS_TRIMMER_RD register. Default is multiply by 1, to keep DLL's 1/4 cycle delay. Integer + fractional formats for multiplier & offset:

OFFS is 2's complement format, w/ bit [4] representing integer

The output of the XFORM is xform_out = (xform_in * MULT + OFFS)/ 16

Largest allowed MULT = 0x17

In the case of overflow, output is clamped according to these rules:

out >= 0x600, xform_out = 0x000

0x600 > out >= 0x400, xform_out = 0x3ff

That means clamping works within 50% above the maximum (0x3ff) dll output or 50% below the minimum (0) dll output.

To make sure that is always satisfied, conservatively program OFFS so that 0x6000 < OFFS < (0x5fff - MULT * 0x3ff)

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

Configure Digital DLL

Offset: 2c0h | Read/Write: R/W | Reset: 0b00000000000000xxx10000

Bit	Reset	Description

Bit	Reset	Description
22:8	0x0	XFORM_DQS_OFFS
4:0	0x10	XFORM_DQS_MULT

15.4.1.80 EMC_DLL_XFORM_QUSE_0

XFORM_QUSE_MULT and XFORM_QUSE_OFFS are a multiplier and offset, respectively that can be used to modify the dll output. Final product may be read out in DQS_TRIMMER_RD register. Default is multiply by 1/2, to give 1/8 cycle delay.

OFFS is 2's complement format, w/ bit [4] representing integer

The output of the XFORM is xform_out = (xform_in * MULT + OFFS)/ 16

Largest allowed MULT = 0x17

In the case of overflow, output is clamped according to these rules:

out >= 0x600, xform_out = 0x000

0x600 > out >= 0x400, xform_out = 0x3ff

That means clamping works within 50% above the maximum (0x3ff) dll output or 50% below the minimum (0) dll output.

To make sure that is always satisfied, conservatively program OFFS so that 0x6000 < OFFS < (0x5fff - MULT * 0x3ff)

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

Configure Digital DLL

Offset: 2c4h | Read/Write: R/W | Reset: 0b000000000000000xxx01000

Bit	Reset	Description
22:8	0x0	XFORM_QUSE_OFFS
4:0	0x8	XFORM_QUSE_MULT

15.4.1.81 EMC_DIG_DLL_UPPER_STATUS_0

Digital DLL Status bits directly from DLL

Digital DLL Status

Offset: 2c8h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15	X	DLL_UPPER_LOCK
14	X	DLL_UPPER_ALARM
13	X	DLL_UPPER_LOCK_TIMEOUT
9:0	X	DLL_UPPER_OUT

15.4.1.82 EMC_DIG_DLL_LOWER_STATUS_0

Digital DLL Status

Offset: 2cch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15	X	DLL_LOWER_LOCK
14	X	DLL_LOWER_ALARM
13	X	DLL_LOWER_LOCK_TIMEOUT
9:0	X	DLL_LOWER_OUT

15.4.1.83 EMC_CTT_TERM_CTRL_0

If CTT_TERMINATION is enabled this controls the strength of the output drivers.

TERM_OVERRIDE forces use of CFG2TMC_CFG*_DRVDN_TERM, CFG2TMC_CFG*_DRVUP_TERM instead (arapb_misc_gp register) of using of AUTO_CAL DRVDN/DRVUP values w/ the following equation applied:

TERM_DRDN = (AUTO_CAL_DRVDN * TERM_SLOPE) / 4 + TERM_OFFSET

TERM_DRUP = (AUTO_CAL_DRVUP * TERM_SLOPE) / 4 + TERM_OFFSET

Recommended values for SLOPE/OFFSET

100ohm CTT: TERM_SLOPE = 0x1, TERM_OFFSET = 0x4

50ohm CTT: TERM_SLOPE = 0x2, TERM_OFFSET = 0x8

Note: setting slope = 0 allows TERM_DRDN/DRVUP = TERM_OFFSET setting slope > 2 is not supported

Boot requirements

- This register (except for fields TERM_DRVDN and TERM_DRVUP) should be parameterized in the BCT and written by the BootROM during coldboot.
- This register (except for fields TERM_DRVDN and TERM_DRVUP) should be saved in the scratch registers and restored by the BootROM during warmboot.

Configure CTT termination output drive strength

Offset: 2dch | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxx01000xxxxx010

Bit	R/W	Reset	Description
31	RW	DISABLED	TERM_OVERRIDE: 0 = DISABLED 1 = ENABLED
28:24	RO	X	TERM_DRVUP
19:15	RO	X	TERM_DRVDN
12:8	RW	0x8	TERM_OFFSET:
2:0	RW	0x2	TERM_SLOPE:

ZQ Calibration Registers -- LPDDR2

Allow an MRW/MRS command to be sent periodically to dram. After ZCAL_REF_CNT, ZCAL_MRW_CMD will be sent to each dram, 1 at a time, followed by ZCAL_WAIT_CNT interval in which no other commands will be allowed to be sent to either dram. So if ZQ_MRW_DEV_SELECTN == 2'b00, it would send the MRW cmd to dev0, wait ZCAL_WAIT_CNT, send cmd to

dev1, wait ZCAL_WAIT_CNT, resume normal operation. It will then wait ZCAL_REF_CNT refreshes before repeating the procedure. ZQ calibration command will not be sent to devices that are 1)unpopulated 2)either in or about to enter either self-refresh or dpd.

Disable this feature by setting ZCAL_REF_CNT.ZCAL_REF_INTERVAL = 0.

15.4.1.84 EMC_ZCAL_REF_CNT_0

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

Configure ZQ Calibration

Offset: 2e0h | Read/Write: R/W | Reset: 0b00000000000000000000000000

Bit	Reset	Description
23:0	0x0	ZCAL_REF_INTERVAL: Number of refreshes to wait between issuance of ZCAL_MRW_CMD. If 0, ZCAL is disabled and internal counter will be reset.

15.4.1.85 EMC_ZCAL_WAIT_CNT_0

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

Configure ZQ Calibration

Offset: 2e4h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7:0	0x0	ZCAL_WAIT_CNT: Number of emc clocks to wait before issuing any commands after sending ZCAL_MRW_CMD.

15.4.1.86 EMC_ZCAL_MRW_CMD_0

This register is shadowed.

Boot requirements

- This register should be parameterized in the BCT and written by the BootROM during coldboot.
- This register should be saved in the scratch registers and restored by the BootROM during warmboot.

Configure ZQ Calibration

Offset: 2e8h | Read/Write: WO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:30	X	ZQ_MRW_DEV_SELECTN: active-low chip-select, 0x0 applies command to both devices (will happen 1 at a time), 0x2 to for only dev0, 0x1 for dev1.
23:16	X	ZQ_MRW_MA: MRW MA field to be sent after ZCAL_REF_CNT

Bit	Reset	Description
7:0	X	ZQ_MRW_OP: MRW OP field to be sent after ZCAL_REF_CNT

15.4.2 MC Registers

15.4.2.1 MC_EMEM_CFG_0

External memory size (in Kbytes): EMEM_SIZE_KB must be <= (Device size in KB * Number of Devices).

This is used to check for DECERR in HW. Only chip spec values are valid. This register needs to be programmed for proper operation.

External memory configuration Register

Offset: 00ch | Read/Write: R/W | Reset: 0b00000100000000000000000

Bit	Reset	Description
21:0	0x10000	EMEM_SIZE_KB

15.4.2.2 MC_EMEM_ADR_CFG_0

This is used to specify the DRAM parameters. MC will use these parameters to derive device, row, bank, column values to EMC.

External memory address configuration Register

Offset: 010h | Read/Write: R/W | Reset: 0b00xxxx0100xxxxxx10xxxxx010

Bit	Reset	Description
25:24	N1	EMEM_NUMDEV: 0 = N1 1 = N2
18:16	D64 MB	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 8 = D1024 MB 8 = D1 GB
9:8	W2	EMEM_BANKWIDTH: 2 = W2 3 = W3
2:0	W9	EMEM_COLWIDTH: 0 = W7 1 = W8 2 = W9 3 = W10 4 = W11

15.4.2.3 MC_EMEM_ARB_CFG0_0

The read/write and write/read counter thresholds are used to generate the read/write switch for the DRAM arbiter when compared to the read/write counter. The counter is incremented every cycle and cleared when the arbitration winner changes from read to write or write to read.

Same-Page consecutive transactions maximum count limits the maximum number of consecutive same-page-based wins the arbiter will grant to the same client before masking that client's same-page indication for one arbitration cycle.

The value of this register is added to EMEM_SP_MAX_GRANT to ensure that the gross-masking cannot interfere with the bank-contention masking. Value of 0 disables. If not disabled, value should be greater than 1.

The no block by bank when high-priority is requesting field is used to disable logic that blocks, by bank, non-HP requests from arbitrating when an HP request is pending. (In other words, when this bit is ENABLED, the logic that blocks non-HP requests is DISABLED.)

Boot requirements: This arbitration configuration register should be parameterized in the BCT and written by the OS during coldboot and warmboot.

External memory arbiter configuration 0

Offset: 014h | Read/Write: R/W | Reset: 0b000xxxxxx01000000100000000110000

Bit	Reset	Description
30	DISABLE	EMEM_NOBLOCK_BY_BANK_WHEN_HP: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
29	DISABLE	EMEM_CLEAR_AP_PREV_SPREQ: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
28	DISABLE	EMEM_CLEAR_SP_ON_AUTOPC: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
21:16	0x10	EMEM_SP_MAX_GRANT_OVERALL
15:8	0x20	EMEM_WRCNT_TH
7:0	0x30	EMEM_RWCNT_TH

15.4.2.4 MC_EMEM_ARB_CFG1_0

The read/write client priority masks are used to mask the read/write winner client priority vector before comparison. The priority vector is made of the following fields:

- the 3 most-significant bits of the priority vector from arbitration
- whether this winner is in the same page
- a bit indicating whether the read/write window has expired

Which are computed into:

priwin = {hp ? 3'h7 : ({timeout,bw}+fpri[1:0])}

privec = {priwin[2:1], spwin, priwin[0], winexpire}

for the purposes of this mask

The lowest significant bit is controlled by the EMEM_WRSWITCH_WWINEXPIRED and EMEM_RWSWITCH_RWINEXPIRED fields described below.

The no-read/write (write/read) switch on bank-blocked requests fields are used to allow/disallow read/write switching when there are requests of the current window type blocked by the bank counters.

The read/write switch is used to make the arbiter switch from reads to writes once the read window has expired and there are read and write requests with equal priority. Otherwise the arbiter would stick with reads as it is better for DRAM bandwidth utilization. Switching is usually the desired behavior.

The write/read switch is used to make the arbiter switch from writes to reads once the write window has expired and there are read and write requests with equal priority. Otherwise the arbiter would stick with writes as it is better for DRAM bandwidth utilization. Switching is usually the desired behavior.

Same-Page consecutive transactions maximum count limits, when bank-contention is detected among two or more clients, the maximum number of consecutive same-page-based wins the arbiter will grant to the same client before masking that client's same-page indication for one arbitration cycle.

Value of 0 disables.

Value of 16 approximates (in certain situations) the behavior of earlier architectures where bank==adr[7:6].

Same-page consecutive transactions maximum count for read/write arbiter limits the maximum number of consecutive same-page-based wins the r/w arbiter will grant after the r/w window has expired before masking the expired window's spwin bit from the r/w arbiter priority vector.

Value of 0 disables.

Boot requirements: This arbitration configuration register should be parameterized in the BCT and written by the OS during coldboot and warmboot.

External memory arbiter configuration 1

Offset: 018h | Read/Write: R/W | Reset: 0b010000xx0100001111x11111x11111

Bit	Reset	Description
29:24	0x10	EMEM_SP_MAX_GRANT_RW
21:16	0x10	EMEM_SP_MAX_GRANT
15	ENABLE	EMEM_WRSWITCH_WWINEXPIRED: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
14	ENABLE	EMEM_RWSWITCH_RWINEXPIRED: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED

Bit	Reset	Description
13	DISABLE	EMEM_NOWRSWITCH_BKBLOCK: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
12	DISABLE	EMEM_NORWSWITCH_BKBLOCK: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
10:6	ALL	EMEM_WCL_MASK: 0 = NONE 63 = ALL
4:0	ALL	EMEM_RCL_MASK: 0 = NONE 63 = ALL

15.4.2.5 MC_EMEM_ARB_CFG2_0

The bank counter threshold is used to block non same-page requests until the number of cycles since the last arbitration bank winner was a (read|write) to the requested bank has reached the threshold. This should mimic tRAS.

The non same-page bank counter threshold is used to block non same-page requests until the number of cycles since the last arbitration bank winner was a (read|write) to the requested bank and it changed the same page, has reached the threshold.

The bank counter thresholds are incremented only when the rest of the memory controller pipeline after the arbiter is ready, so that they are kept in sync with the requests sent to the external memory controller. This should mimic (tRAS+tRP).

There is a one-cycle imprecision on number of cycles related to the bank counter thresholds.

Boot requirements: This arbitration configuration register should be parameterized in the BCT and written by the OS during coldboot and warmboot.

External memory arbiter config 2

Offset: 01ch | Read/Write: R/W | Reset: 0b001100xx001000xx001100xx001000

Bit	Reset	Description
29:24	0xc	EMEM_BANKCNT_NSP_WR_TH
21:16	0x8	EMEM_BANKCNT_WR_TH
13:8	0xc	EMEM_BANKCNT_NSP_RD_TH
5:0	0x8	EMEM_BANKCNT_RD_TH

15.4.2.6 MC_GART_CONFIG_0

GART_CONFIG contains a single bit enable for the GART functionality. When DISABLED, the GART passes through addresses without modification. Because the address-bound check occurs after GART translation, an unmodified GART address will be treated as-out-of-bounds by the MC.

When ENABLED, the GART remaps addresses as specified by the GART entries.

Offset: 024h | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
0	DISABLE	GART_ENABLE: 0 = DISABLE 1 = ENABLE

15.4.2.7 MC_GART_ENTRY_ADDR_0

GART_ENTRY_ADDR is one of two registers used to access the GART table. The GART_ENTRY_ADDR value defines which entry will be read or written by accesses to the GART_ENTRY_DATA register.

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

Bit	Reset	Description
24:12	X	GART_ENTRY_ADDR_TABLE_ADDR

15.4.2.8 MC_GART_ENTRY_DATA_0

GART_ENTRY_DATA is one of two registers used to access the GART table. A read of the GART_ENTRY_DATA register returns the GART entry addressed by GART_ENTRY_ADDR.

Similarly, a write to the GART_ENTRY_DATA register updates the GART entry addressed by GART_ENTRY_ADDR.

Offset: 02ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	GART_ENTRY_DATA_PHYS_ADDR_VALID
30:12	X	GART_ENTRY_DATA_PHYS_ADDR

15.4.2.9 MC_GART_ERROR_REQ_0

When any client attempts to access an invalid GART page for reading or writing, an INVALID_GART_PAGE_INT occurs and the attributes of the first such request are stored in GART_ERROR_REQ.

The global memory client IDs are defined as follows:

Table 51 Global Memory Client IDs

CLIENT	ID	UNIT
cbr_display0a	0	dc
cbr_display0ab	1	dcb
cbr_display0b	2	dc
cbr_display0bb	3	dcb
cbr_display0c	4	dc
cbr_display0cb	5	dcb

CLIENT	ID	UNIT
cbr_display1b	6	dc
cbr_display1bb	7	dcb
cbr_eppup	8	epp
cbr_g2pr	9	g2
cbr_g2sr	10	g2
cbr_mpeunifbr	11	mpeb
cbr_viruv	12	vi
csr_avpcarm7r	13	avpc
csr_displayhc	14	dc
csr_displayhcb	15	dcb
csr_fdcdrd	16	nv
csr_g2dr	17	g2
csr_host1xdmar	18	hc
csr_host1xr	19	hc
csr_idxsrd	20	nv
csr_mpcorer	21	cpu
csr_mpe_ipred	22	mpeb
csr_mpeamemrd	23	mpea
csr_mpecsrd	24	mpec
csr_ppcsahbdmar	25	ppcs
csr_ppcsahbslvr	26	ppcs
csr_texsrd	27	nv
csr_vdebsevr	28	vde
csr_vdember	29	vde
csr_vdemcer	30	vde
csr_vdetper	31	vde
cbw_eppu	32	epp

CLIENT	ID	UNIT
cbw_eppv	33	epp
cbw_eppy	34	epp
cbw_mpeunifbw	35	mpeb
cbw_viwsb	36	vi
cbw_viwu	37	vi
cbw_viwv	38	vi
cbw_viwy	39	vi
ccw_g2dw	40	g2
csw_avpcarm7w	41	avpc
csw_fdcdwr	42	nv
csw_host1xw	43	hc
csw_ispw	44	isp
csw_mpcorew	45	CPU
csw_mpecswr	46	mpec
csw_ppcsahbdmaw	47	ppcs
csw_ppcsahbslvw	48	ppcs
csw_vdebsevw	49	vde
csw_vdembew	50	vde
csw_vdetpmw	51	vde

Offset: 030h | Read/Write: RO | Reset: 0bxxxxxxx

Bit	Reset	Description
6:1	X	GART_ERROR_CLIENT_ID
0	X	GART_ERROR_DIRECTION: 0 = READ 1 = WRITE

15.4.2.10 MC_GART_ERROR_ADDR_0

When any client attempts to access an invalid GART page for reading or writing, an INVALID_GART_PAGE_INT occurs and the address of the first such request are stored in GART_ERROR_ADDR.

Offset: 034h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	GART_ERROR_ADDRESS

15.4.2.11 MC_TIMEOUT_CTRL_0

Time-out clock counter scale factors. Reset value is NV_MC_TM_SFACTOR_RESET. A value of zero means that the time-out clock counter is disabled.

A value of N (non-zero) means that the time-out clock counter is decremented every 2^(N+1) memory cycles. This is used in conjunction with the _TMVAL registers defined below.

Time-out control Register

Offset: 03ch | Read/Write: R/W | Reset: 0b0101

Bit	Reset	Description
6	FROM_CIF_FIFO	TMCREDITS: 0 = FROM_CIF_FIFO 1 = ONE
5:3	0x5	EMEM_TM_SFACTOR

15.4.2.12 MC_DECERR_EMEM_OTHERS_STATUS_0

If a decode error from graphics is detected for EMEM, this register records information about this access: read/write indicator and request ID (global client ID assigned by the memory controller).

The values remain unchanged until the interrupt is cleared.

The global memory client IDs are defined as given in Global Memory Client IDs

EMEM Decode Error from graphics status Register

Offset: 058h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	DECERR_EMEM_OTHERS_RW: 0 = READ 1 = WRITE
5:0	X	DECERR_EMEM_OTHERS_ID

15.4.2.13 MC_DECERR_EMEM_OTHERS_ADR_0

If a decode error from graphics is detected for EMEM, this register records the request byte address.

The value remains unchanged until the interrupt is cleared.

EMEM Decode Error from graphics Register

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

Bit	Reset	Description
31:0	X	DECERR_EMEM_OTHERS_ADR

15.4.2.14 MC_CLIENT_CTRL_0

Active high. Reset value is enabled. When enabled, the client requests are free to go to arbitration. When disabled, the client requests are blocked before arbitration.

Memory Client Control Register

Offset: 100h | Read/Write: R/W | Reset: 0b111111111111111

Bit	Reset	Description
14	ENABLE	VI_ENABLE: 0 = DISABLE 1 = ENABLE
13	ENABLE	VDE_ENABLE: 0 = DISABLE 1 = ENABLE
12	ENABLE	PPCS_ENABLE: 0 = DISABLE 1 = ENABLE
11	ENABLE	NV_ENABLE: 0 = DISABLE 1 = ENABLE
10	ENABLE	MPEC_ENABLE: 0 = DISABLE 1 = ENABLE
9	ENABLE	MPEB_ENABLE: 0 = DISABLE 1 = ENABLE
8	ENABLE	MPEA_ENABLE: 0 = DISABLE 1 = ENABLE
7	ENABLE	MPCORE_ENABLE: 0 = DISABLE 1 = ENABLE
6	ENABLE	ISP_ENABLE: 0 = DISABLE 1 = ENABLE
5	ENABLE	HC_ENABLE: 0 = DISABLE 1 = ENABLE
4	ENABLE	G2_ENABLE: 0 = DISABLE 1 = ENABLE
3	ENABLE	EPP_ENABLE: 0 = DISABLE 1 = ENABLE
2	ENABLE	DCB_ENABLE: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1	ENABLE	DC_ENABLE: 0 = DISABLE 1 = ENABLE
0	ENABLE	AVPC_ENABLE: 0 = DISABLE 1 = ENABLE

15.4.2.15 MC_CLIENT_HOTRESETN_0

Active low. Reset value is no hot reset. When enabled, the client requests before arbitration are cleared. A proper client reset sequence is as followed:

- Clear client bit in the MC_CLIENT_CTRL register to block its requests.
- Poll the C_OUTREQCNT register till zero.
- Clear module bit in the APBDEV_RSTREG register to reset the module, and clear client bit in the MC_CLIENT_HOTRESETN register to clear the client requests sitting before arbitration.
- Set client bit in the MC_CLIENT_HOTRESETN register to release the hot reset.
- Set module bit in the APBDEV_RSTREG register to release the module reset.
- Set client bit in the MC_CLIENT_CTRL register to allow new requests to proceed to arbitration.

Memory Client Hot Reset Register

Offset: 104h| Read/Write: R/W | Reset: 0b111111111111111

Bit	Reset	Description
14	DISABLE	VI_HOTRESETN: 0 = ENABLE 1 = DISABLE
13	DISABLE	VDE_HOTRESETN: 0 = ENABLE 1 = DISABLE
12	DISABLE	PPCS_HOTRESETN: 0 = ENABLE 1 = DISABLE
11	DISABLE	NV_HOTRESETN: 0= ENABLE 1 = DISABLE
10	DISABLE	MPEC_HOTRESETN: 0 = ENABLE 1 = DISABLE
9	DISABLE	MPEB_HOTRESETN: 0 = ENABLE 1 = DISABLE
8	DISABLE	MPEA_HOTRESETN: 0 = ENABLE 1 = DISABLE

Bit	Reset	Description
7	DISABLE	MPCORE_HOTRESETN: 0 = ENABLE 1 = DISABLE
6	DISABLE	ISP_HOTRESETN: 0 = ENABLE 1 = DISABLE
5	DISABLE	HC_HOTRESETN: 0 = ENABLE 1 = DISABLE
4	DISABLE	G2_HOTRESETN: 0 = ENABLE 1 = DISABLE
3	DISABLE	EPP_HOTRESETN: 0 = ENABLE 1 = DISABLE
2	DISABLE	DCB_HOTRESETN: 0 = ENABLE 1 = DISABLE
1	DISABLE	DC_HOTRESETN: 0 = ENABLE 1 = DISABLE
0	DISABLE	AVPC_HOTRESETN: 0 = ENABLE 1 = DISABLE

15.4.2.16 MC_LOWLATENCY_RAWLOGIC_WRITE_PARTICIPANTS_0

When ENABLED, the write client will participate in the low-latency RAW-protection logic.

By default, all clients participate. When a client is participating in the RAW logic, if that client has arbitrated a write to bank K, and while that write is arbitrated in MC but not in EMC, and if a low-latency read arrives, also to bank K, that low-latency read will be stalled until the write is committed to EMC.

Low-latency client RAW-protection logic

Offset: 110h | Read/Write: R/W | Reset: 0b11111111111111111111

Bit	Reset	Description
19	ENABLE	VDETPMW_LL_RAW_PARTICIPATE: 0 = DISABLE 1 = ENABLE
18	ENABLE	VDEMBEW_LL_RAW_PARTICIPATE: 0 = DISABLE 1 = ENABLE
17	ENABLE	VDEBSEVW_LL_RAW_PARTICIPATE: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
16	ENABLE	PPCSAHBSLVW_LL_RAW_PARTICIPATE: 0 = DISABLE 1 = ENABLE
15	ENABLE	PPCSAHBDMAW_LL_RAW_PARTICIPATE: 0 = DISABLE 1 = ENABLE
14	ENABLE	MPECSWR_LL_RAW_PARTICIPATE: 0 = DISABLE 1 = ENABLE
13	ENABLE	MPCOREW_LL_RAW_PARTICIPATE: 0 = DISABLE 1 = ENABLE
12	ENABLE	ISPW_LL_RAW_PARTICIPATE: 0 = DISABLE 1 = ENABLE
11	ENABLE	HOST1XW_LL_RAW_PARTICIPATE: 0 = DISABLE 1 = ENABLE
10	ENABLE	FDCDWR_LL_RAW_PARTICIPATE: 0 = DISABLE 1 = ENABLE
9	ENABLE	AVPCARM7W_LL_RAW_PARTICIPATE 0 = DISABLE 1 = ENABLE
8	ENABLE	G2DW_LL_RAW_PARTICIPATE: 0 = DISABLE 1 = ENABLE
7	ENABLE	VIWY_LL_RAW_PARTICIPATE: 0 = DISABLE 1 = ENABLE
6	ENABLE	VIWV_LL_RAW_PARTICIPATE: 0 = DISABLE 1 = ENABLE
5	ENABLE	VIWU_LL_RAW_PARTICIPATE: 0 = DISABLE 1 = ENABLE
4	ENABLE	VIWSB_LL_RAW_PARTICIPATE: 0 = DISABLE 1 = ENABLE
3	ENABLE	MPEUNIFBW_LL_RAW_PARTICIPATE: 0 = DISABLE 1 = ENABLE
2	ENABLE	EPPY_LL_RAW_PARTICIPATE: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1	ENABLE	EPPV_LL_RAW_PARTICIPATE: 0 = DISABLE 1 = ENABLE
0	ENABLE	EPPU_LL_RAW_PARTICIPATE: 0 = DISABLE 1 = ENABLE

15.4.2.17 MC_BWSHARE_TMVAL_0

A value of zero means that the timer counter is disabled. A value of N (non-zero) means that the BW Share increment happens every 2^(N+1) memory cycles.

Bandwidth share increment timer value

Offset: 114h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7:4	0x0	BW_TM_SFACTOR2
3:0	0x0	BW_TM_SFACTOR1

15.4.2.18 MC_BWSHARE_EMEM_CTRL_0_0

Reset value is disabled. Controls whether a client should participate in EMEM Bandwidth Share mechanism. Bandwidth share is per super client. If Enabled, make sure the super client's bandwidth share registers are programmed.

Memory Client EMEM Bandwidth Share control register 0

Offset: 120h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	DISABLE	VDETPER_BW_EMEM 0 = DISABLE 1 = ENABLE
30	DISABLE	VDEMCER_BW_EMEM: 0 = DISABLE 1 = ENABLE
29	DISABLE	VDEMBER_BW_EMEM: 0 = DISABLE 1 = ENABLE
28	DISABLE	VDEBSEVR_BW_EMEM 0 = DISABLE 1 = ENABLE
27	DISABLE	TEXSRD_BW_EMEM: 0 = DISABLE 1 = ENABLE
26	DISABLE	PPCSAHBSLVR_BW_EMEM: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
25	DISABLE	PPCSAHBDMAR_BW_EMEM: 0 = DISABLE 1 = ENABLE
24	DISABLE	MPECSRD_BW_EMEM: 0 = DISABLE 1 = ENABLE
23	DISABLE	MPEAMEMRD_BW_EMEM: 0 = DISABLE 1 = ENABLE
22	DISABLE	MPE_IPRED_BW_EMEM: 0 = DISABLE 1 = ENABLE
21	DISABLE	MPCORER_BW_EMEM: 0 = DISABLE 1 = ENABLE
20	DISABLE	IDXSRD_BW_EMEM: 0 = DISABLE 1 = ENABLE
19	DISABLE	HOST1XR_BW_EMEM: 0 = DISABLE 1 = ENABLE
18	DISABLE	HOST1XDMAR_BW_EMEM: 0 = DISABLE 1 = ENABLE
17	DISABLE	G2DR_BW_EMEM: 0 = DISABLE 1 = ENABLE
16	DISABLE	FDCDRD_BW_EMEM: 0 = DISABLE 1 = ENABLE
15	DISABLE	DISPLAYHCB_BW_EMEM: 0 = DISABLE 1 = ENABLE
14	DISABLE	DISPLAYHC_BW_EMEM: 0 = DISABLE 1 = ENABLE
13	DISABLE	AVPCARM7R_BW_EMEM 0 = DISABLE 1 = ENABLE
12	DISABLE	VIRUV_BW_EMEM: 0 = DISABLE 1 = ENABLE
11	DISABLE	MPEUNIFBR_BW_EMEM: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
10	DISABLE	G2SR_BW_EMEM: 0 = DISABLE 1 = ENABLE
9	DISABLE	G2PR_BW_EMEM: 0 = DISABLE 1 = ENABLE
8	DISABLE	EPPUP_BW_EMEM: 0 = DISABLE 1 = ENABLE
7	DISABLE	DISPLAY1BB_BW_EMEM: 0 = DISABLE 1 = ENABLE
6	DISABLE	DISPLAY1B_BW_EMEM: 0 = DISABLE 1 = ENABLE
5	DISABLE	DISPLAY0CB_BW_EMEM: 0 = DISABLE 1 = ENABLE
4	DISABLE	DISPLAY0C_BW_EMEM: 0 = DISABLE 1 = ENABLE
3	DISABLE	DISPLAY0BB_BW_EMEM: 0 = DISABLE 1 = ENABLE
2	DISABLE	DISPLAY0B_BW_EMEM: 0 = DISABLE 1 = ENABLE
1	DISABLE	DISPLAY0AB_BW_EMEM: 0 = DISABLE 1 = ENABLE
0	DISABLE	DISPLAY0A_BW_EMEM: 0 = DISABLE 1 = ENABLE

15.4.2.19 MC_BWSHARE_EMEM_CTRL_1_0

Reset value is disabled. Controls whether a client should participate in EMEM Bandwidth Share mechanism. Bandwidth share is per super client. If Enabled, make sure the super client's bandwidth share registers are programmed.

Memory Client EMEM Bandwidth Share control register 1

Offset: 124h | Read/Write: R/W | Reset: 0b00000000000000000000000000

Bit	Reset	Description
19	DISABLE	VDETPMW_BW_EMEM: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
18	DISABLE	VDEMBEW_BW_EMEM: 0 = DISABLE 1 = ENABLE
17	DISABLE	VDEBSEVW_BW_EMEM: 0 = DISABLE 1 = ENABLE
16	DISABLE	PPCSAHBSLVW_BW_EMEM: 0 = DISABLE 1 = ENABLE
15	DISABLE	PPCSAHBDMAW_BW_EMEM: 0 = DISABLE 1 = ENABLE
14	DISABLE	MPECSWR_BW_EMEM: 0 = DISABLE 1 = ENABLE
13	DISABLE	MPCOREW_BW_EMEM: 0 = DISABLE 1 = ENABLE
12	DISABLE	ISPW_BW_EMEM: 0 = DISABLE 1 = ENABLE
11	DISABLE	HOST1XW_BW_EMEM: 0 = DISABLE 1 = ENABLE
10	DISABLE	FDCDWR_BW_EMEM: 0 = DISABLE 1 = ENABLE
9	DISABLE	AVPCARM7W_BW_EMEM 0 = DISABLE 1 = ENABLE
8	DISABLE	G2DW_BW_EMEM: 0 = DISABLE 1 = ENABLE
7	DISABLE	VIWY_BW_EMEM: 0 = DISABLE 1 = ENABLE
6	DISABLE	VIWV_BW_EMEM: 0 = DISABLE 1 = ENABLE
5	DISABLE	VIWU_BW_EMEM: 0 = DISABLE 1 = ENABLE
4	DISABLE	VIWSB_BW_EMEM: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
3	DISABLE	MPEUNIFBW_BW_EMEM: 0 = DISABLE 1 = ENABLE
2	DISABLE	EPPY_BW_EMEM: 0 = DISABLE 1 = ENABLE
1	DISABLE	EPPV_BW_EMEM: 0 = DISABLE 1 = ENABLE
0	DISABLE	EPPU_BW_EMEM: 0 = DISABLE 1 = ENABLE

15.4.2.20 MC_AVPC_ORRC_0

Read only. This reports the number of outstanding client requests after arbitration. It is incremented by HW when a client request is granted by the arbiter. For read clients, it is decremented by HW when a return data is sent back to the client.

AVPC Outstanding Request Register

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

Bit	Reset	Description
7:0	X	AVPC_OUTREQCNT

15.4.2.21 MC_DC_ORRC_0

Read only. This reports the number of outstanding client requests after arbitration. It is incremented by HW when a client request is granted by the arbiter. For read clients, it is decremented by HW when a return data is sent back to the client.

DC Outstanding Request Register

Offset: 144h | Read/Write: RO | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	DC_OUTREQCNT

15.4.2.22 MC_DCB_ORRC_0

Read only. This reports the number of outstanding client requests after arbitration. It is incremented by HW when a client request is granted by the arbiter. For read clients, it is decremented by HW when a return data is sent back to the client.

DCB Outstanding Request Register

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

Bit	Reset	Description
7:0	X	DCB_OUTREQCNT

15.4.2.23 MC_EPP_ORRC_0

Read only. This reports the number of outstanding client requests after arbitration. It is incremented by HW when a client request is granted by the arbiter. For read clients, it is decremented by HW when a return data is sent back to the client.

EPP Outstanding Request Register

Offset: 14ch | Read/Write: RO | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	EPP_OUTREQCNT

15.4.2.24 MC_G2_ORRC_0

Read only. This reports the number of outstanding client requests after arbitration. It is incremented by HW when a client request is granted by the arbiter. For read clients, it is decremented by HW when a return data is sent back to the client.

G2 Outstanding Request Register

Offset: 150h | Read/Write: RO | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	G2_OUTREQCNT

15.4.2.25 MC_HC_ORRC_0

Read only. This reports the number of outstanding client requests after arbitration. It is incremented by HW when a client request is granted by the arbiter. For read clients, it is decremented by HW when a return data is sent back to the client.

HC Outstanding Request Register

Offset: 154h | Read/Write: RO | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	HC_OUTREQCNT

15.4.2.26 MC_ISP_ORRC_0

Read only. This reports the number of outstanding client requests after arbitration. It is incremented by HW when a client request is granted by the arbiter. For read clients, it is decremented by HW when a return data is sent back to the client.

ISP Outstanding Request Register

Offset: 158h | Read/Write: RO | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	ISP_OUTREQCNT

15.4.2.27 MC_MPCORE_ORRC_0

Read only. This reports the number of outstanding client requests after arbitration. It is incremented by HW when a client request is granted by the arbiter. For read clients, it is decremented by HW when a return data is sent back to the client.

CPU Outstanding Request Register

Offset: 15ch | Read/Write: RO | Reset: 0bxxxxxxxx | Default: 0000.0000

Bit	Reset	Description
7:0	X	MPCORE_OUTREQCNT

15.4.2.28 MC_MPEA_ORRC_0

Read only. This reports the number of outstanding client requests after arbitration. It is incremented by HW when a client request is granted by the arbiter. For read clients, it is decremented by HW when a return data is sent back to the client.

MPEA Outstanding Request Register

Offset: 160h | Read/Write: RO | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	MPEA_OUTREQCNT

15.4.2.29 MC_MPEB_ORRC_0

Read only. This reports the number of outstanding client requests after arbitration. It is incremented by HW when a client request is granted by the arbiter. For read clients, it is decremented by HW when a return data is sent back to the client.

MPEB Outstanding Request Register

Offset: 164h | Read/Write: RO | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	MPEB_OUTREQCNT

15.4.2.30 MC_MPEC_ORRC_0

Read only. This reports the number of outstanding client requests after arbitration. It is incremented by HW when a client request is granted by the arbiter. For read clients, it is decremented by HW when a return data is sent back to the client.

MPEC Outstanding Request Register

Offset: 168h | Read/Write: RO | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	MPEC_OUTREQCNT

15.4.2.31 MC_NV_ORRC_0

Read only. This reports the number of outstanding client requests after arbitration. It is incremented by HW when a client request is granted by the arbiter. For read clients, it is decremented by HW when a return data is sent back to the client.

NV Outstanding Request Register

Offset: 16ch | Read/Write: RO | Reset: 0bxxxxxxxx

Bit	Reset	Description

Bit	Reset	Description
7:0	X	NV_OUTREQCNT

15.4.2.32 MC_PPCS_ORRC_0

Read only. This reports the number of outstanding client requests after arbitration. It is incremented by HW when a client request is granted by the arbiter. For read clients, it is decremented by HW when a return data is sent back to the client.

PPCS Outstanding Request Register

Offset: 170h | Read/Write: RO | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	PPCS_OUTREQCNT

15.4.2.33 MC_VDE_ORRC_0

Read only. This reports the number of outstanding client requests after arbitration. It is incremented by HW when a client request is granted by the arbiter. For read clients, it is decremented by HW when a return data is sent back to the client.

VDE Outstanding Request Register

Offset: 174h | Read/Write: RO | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	VDE_OUTREQCNT

15.4.2.34 MC_VI_ORRC_0

Read only. This reports the number of outstanding client requests after arbitration. It is incremented by HW when a client request is granted by the arbiter. For read clients, it is decremented by HW when a return data is sent back to the client.

VI Outstanding Request Register

Offset: 178h | Read/Write: RO | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	VI_OUTREQCNT

15.4.2.35 MC_FPRI_CTRL_AVPC_0

Reset value defaults to LOW for most clients, but may be different for certain clients.

Fixed-priority Register for AVPC clients

Offset: 17ch | Read/Write: R/W | Reset: 0b0101

Bit	Reset	Description
3:2	LOW	AVPCARM7W_PRIVAL0 = LOWEST 1 = LOW 2 = MED 3 = HIGH

Bit	Reset	Description
1:0	LOW	AVPCARM7R_PRIVAL0 = LOWEST 1 = LOW 2 = MED 3 = HIGH

15.4.2.36 MC_FPRI_CTRL_DC_0

Reset value defaults to LOW for most clients, but may be different for certain clients.

Fixed-priority Register for dc clients

Offset: 180h | Read/Write: R/W | Reset: 0b0101010101

Bit	Reset	Description
9:8	LOW	DISPLAYHC_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH
7:6	LOW	DISPLAY1B_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH
5:4	LOW	DISPLAY0C_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH
3:2	LOW	DISPLAY0B_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH
1:0	LOW	DISPLAY0A_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH

15.4.2.37 MC_FPRI_CTRL_DCB_0

Reset value defaults to LOW for most clients, but may be different for certain clients

Fixed-priority Register for dcb clients

Offset: 184h | Read/Write: R/W | Reset: 0b0101010101

Bit	Reset	Description

Bit	Reset	Description
9:8	LOW	DISPLAYHCB_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH
7:6	LOW	DISPLAY1BB_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH
5:4	LOW	DISPLAY0CB_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH
3:2	LOW	DISPLAY0BB_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH
1:0	LOW	DISPLAY0AB_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH

15.4.2.38 MC_FPRI_CTRL_EPP_0

Reset value defaults to LOW for most clients, but may be different for certain clients

Fixed-priority Register for epp clients

Offset: 188h | Read/Write: R/W | Reset: 0b01010101

Bit	Reset	Description
7:6	LOW	EPPY_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH
5:4	LOW	EPPV_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH
3:2	LOW	EPPU_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH

Bit	Reset	Description
1:0	LOW	EPPUP_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH

15.4.2.39 MC_FPRI_CTRL_G2_0

Reset value defaults to LOW for most clients, but may be different for certain clients

Fixed-priority Register for g2 clients

Offset: 18ch | Read/Write: R/W | Reset: 0b00000000 | Default: 0000.0000

Bit	Reset	Description
7:6	LOWEST	G2DW_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH
5:4	LOWEST	G2DR_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH
3:2	LOWEST	G2SR_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH
1:0	LOWEST	G2PR_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH

15.4.2.40 MC_FPRI_CTRL_HC_0

Reset value defaults to LOW for most clients, but may be different for certain clients.

Fixed-priority Register for hc clients

Offset: 190h | Read/Write: R/W | Reset: 0b010101

Bit	Reset	Description
5:4	LOW	HOST1XW_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH

Bit	Reset	Description
3:2	LOW	HOST1XR_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH
1:0	LOW	HOST1XDMAR_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH

15.4.2.41 MC_FPRI_CTRL_ISP_0

Reset value defaults to LOW for most clients, but may be different for certain clients.

Fixed-priority Register for isp clients

Offset: 194h | Read/Write: R/W | Reset: 0b01

Bit	Reset	Description
1:0	LOW	ISPW_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH

15.4.2.42 MC_FPRI_CTRL_MPCORE_0

Reset value defaults to LOW for most clients, but may be different for certain clients.

Fixed-priority Register for CPU clients

Offset: 198h | Read/Write: R/W | Reset: 0b0101

Bit	Reset	Description
3:2	LOW	MPCOREW_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH
1:0	LOW	MPCORER_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH

15.4.2.43 MC_FPRI_CTRL_MPEA_0

Reset value defaults to LOW for most clients, but may be different for certain clients.

Fixed-priority Register for mpea clients

Offset: 19ch | Read/Write: R/W | Reset: 0b01

Bit	Reset	Description
1:0	LOW	MPEAMEMRD_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH

15.4.2.44 MC_FPRI_CTRL_MPEB_0

Reset value defaults to LOW for most clients, but may be different for certain clients.

Fixed-priority Register for mpeb clients

Offset: 1a0h | Read/Write: R/W | Reset: 0b010101

Bit	Reset	Description
5:4	LOW	MPEUNIFBW_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH
3:2	LOW	MPE_IPRED_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH
1:0	LOW	MPEUNIFBR_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH

15.4.2.45 MC_FPRI_CTRL_MPEC_0

Reset value defaults to LOW for most clients, but may be different for certain clients.

Fixed-priority Register for mpec clients

Offset: 1a4h | Read/Write: R/W | Reset: 0b0101

Bit	Reset	Description
3:2	LOW	MPECSWR_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH

Bit	Reset	Description
1:0	LOW	MPECSRD_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH

15.4.2.46 MC_FPRI_CTRL_NV_0

Reset value defaults to LOW for most clients, but may be different for certain clients.

Fixed-priority Register for nv clients

Offset: 1a8h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7:6	LOWEST	FDCDWR_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH
5:4	LOWEST	TEXSRD_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH
3:2	LOWEST	IDXSRD_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH
1:0	LOWEST	FDCDRD_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH

15.4.2.47 MC_FPRI_CTRL_PPCS_0

Reset value defaults to LOW for most clients, but may be different for certain clients.

Fixed-priority Register for ppcs clients

Offset: 1ach | Read/Write: R/W | Reset: 0b01010101

Bit	Reset	Description
7:6	LOW	PPCSAHBSLVW_PRIVAL0 = LOWEST 1 = LOW 2 = MED 3 = HIGH

Bit	Reset	Description
5:4	LOW	PPCSAHBDMAW_PRIVAL0 = LOWEST 1 = LOW 2 = MED 3 = HIGH
3:2	LOW	PPCSAHBSLVR_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH
1:0	LOW	PPCSAHBDMAR_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH

15.4.2.48 MC_FPRI_CTRL_VDE_0

Reset value defaults to LOW for most clients, but may be different for certain clients.

Fixed-priority Register for vde clients

Offset: 1b0h | Read/Write: R/W | Reset: 0b01010101010101

Bit	Reset	Description
13:12	LOW	VDETPMW_PRIVAL 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH
11:10	LOW	VDEMBEW_PRIVAL 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH
9:8	LOW	VDEBSEVW_PRIVAL 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH
7:6	LOW	VDETPER_PRIVAL 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH
5:4	LOW	VDEMCER_PRIVAL 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH

Bit	Reset	Description
3:2	LOW	VDEMBER_PRIVAL 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH
1:0	LOW	VDEBSEVR_PRIVAL 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH

15.4.2.49 MC_FPRI_CTRL_VI_0

Reset value defaults to LOW for most clients, but may be different for certain clients.

Fixed-priority Register for vi clients

Offset: 1b4h | Read/Write: R/W | Reset: 0b0101010101

Bit	Reset	Description
9:8	LOW	VIWY_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH
7:6	LOW	VIWV_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH
5:4	LOW	VIWU_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH
3:2	LOW	VIWSB_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH
1:0	LOW	VIRUV_PRIVAL: 0 = LOWEST 1 = LOW 2 = MED 3 = HIGH

15.4.2.50 MC_TIMEOUT_AVPC_0

Reset value defaults to 8 for most clients, but may be different for certain clients If not zero, the client request time-out counter is reloaded with this value when it reaches zero. It is decremented when there is a pending request and the time-out clock counter for the request destination (internal/external memory) reaches zero.

When the client request time-out counter reaches zero, a number of time-out credits equal to the number of its requests pending inside the memory controller client FIFO are allocated to this client.

The number of time-out credits is decremented each time a request is granted. As long as there are credits, the client time-out priority bit is set.

Request time-out Register for AVPC clients

Offset: 1b8h | Read/Write: R/W | Reset: 0b10001000

Bit	Reset	Description
7:4	0x8	AVPCARM7W_TMVAL
3:0	0x8	AVPCARM7R_TMVAL

15.4.2.51 MC_TIMEOUT_DC_0

Reset value defaults to 8 for most clients, but may be different for certain clients. If not zero, the client request time-out counter is reloaded with this value when it reaches zero. It is decremented when there is a pending request and the time-out clock counter for the request destination (internal/external memory) reaches zero.

When the client request time-out counter reaches zero, a number of time-out credits equal to the number of its requests pending inside the memory controller client FIFO are allocated to this client.

The number of time-out credits is decremented each time a request is granted. As long as there are credits, the client time-out priority bit is set.

Request time-out Register for dc clients

Offset: 1bch | Read/Write: R/W | Reset: 0b10001000100010001000

Bit	Reset	Description
19:16	0x8	DISPLAYHC_TMVAL
15:12	0x8	DISPLAY1B_TMVAL
11:8	0x8	DISPLAY0C_TMVAL
7:4	0x8	DISPLAY0B_TMVAL
3:0	0x8	DISPLAY0A_TMVAL

15.4.2.52 MC_TIMEOUT_DCB_0

Reset value defaults to 8 for most clients, but may be different for certain clients. If not zero, the client request time-out counter is reloaded with this value when it reaches zero. It is decremented when there is a pending request and the time-out clock counter for the request destination (internal/external memory) reaches zero.

When the client request time-out counter reaches zero, a number of time-out credits equal to the number of its requests pending inside the memory controller client FIFO are allocated to this client.

The number of time-out credits is decremented each time a request is granted. As long as there are credits, the client time-out priority bit is set.

Request time-out Register for dcb clients

Offset: 1c0h | Read/Write: R/W | Reset: 0b10001000100010001000

Bit	Reset	Description
19:16	0x8	DISPLAYHCB_TMVAL
15:12	0x8	DISPLAY1BB_TMVAL
11:8	0x8	DISPLAY0CB_TMVAL
7:4	0x8	DISPLAY0BB_TMVAL
3:0	0x8	DISPLAY0AB_TMVAL

15.4.2.53 MC_TIMEOUT_EPP_0

Reset value defaults to 8 for most clients, but may be different for certain clients. If not zero, the client request time-out counter is reloaded with this value when it reaches zero. It is decremented when there is a pending request and the time-out clock counter for the request destination (internal/external memory) reaches zero.

When the client request time-out counter reaches zero, a number of time-out credits equal to the number of its requests pending inside the memory controller client FIFO are allocated to this client.

The number of time-out credits is decremented each time a request is granted. As long as there are credits, the client time-out priority bit is set.

Request time-out Register for epp clients

Offset: 1c4h | Read/Write: R/W | Reset: 0b1000100010001000

Bit	Reset	Description
15:12	0x8	EPPY_TMVAL
11:8	0x8	EPPV_TMVAL
7:4	0x8	EPPU_TMVAL
3:0	0x8	EPPUP_TMVAL

15.4.2.54 MC_TIMEOUT_G2_0

Reset value defaults to 8 for most clients, but may be different for certain clients. If not zero, the client request time-out counter is reloaded with this value when it reaches zero. It is decremented when there is a pending request and the time-out clock counter for the request destination (internal/external memory) reaches zero.

When the client request time-out counter reaches zero, a number of time-out credits equal to the number of its requests pending inside the memory controller client FIFO are allocated to this client.

The number of time-out credits is decremented each time a request is granted. As long as there are credits, the client time-out priority bit is set.

Request time-out Register for g2 clients

Offset: 1c8h | Read/Write: R/W | Reset: 0b100010001000100

Bit	Reset	Description
15:12	0x8	G2DW_TMVAL
11:8	0x8	G2DR_TMVAL

Bit	Reset	Description
7:4	0x8	G2SR_TMVAL
3:0	0x8	G2PR_TMVAL

15.4.2.55 MC_TIMEOUT_HC_0

Reset value defaults to 8 for most clients, but may be different for certain clients. If not zero, the client request time-out counter is reloaded with this value when it reaches zero. It is decremented when there is a pending request and the time-out clock counter for the request destination (internal/external memory) reaches zero.

When the client request time-out counter reaches zero, a number of time-out credits equal to the number of its requests pending inside the memory controller client FIFO are allocated to this client.

The number of time-out credits is decremented each time a request is granted. As long as there are credits, the client time-out priority bit is set.

Request time-out Register for hc clients

Offset: 1cch | Read/Write: R/W | Reset: 0b100010001000

Bit	Reset	Description
11:8	0x8	HOST1XW_TMVAL
7:4	0x8	HOST1XR_TMVAL
3:0	0x8	HOST1XDMAR_TMVAL

15.4.2.56 MC_TIMEOUT_ISP_0

Reset value defaults to 8 for most clients, but may be different for certain clients. If not zero, the client request time-out counter is reloaded with this value when it reaches zero. It is decremented when there is a pending request and the time-out clock counter for the request destination (internal/external memory) reaches zero.

When the client request time-out counter reaches zero, a number of time-out credits equal to the number of its requests pending inside the memory controller client FIFO are allocated to this client.

The number of time-out credits is decremented each time a request is granted. As long as there are credits, the client time-out priority bit is set.

Request time-out Register for isp clients

Offset: 1d0h | Read/Write: R/W | Reset: 0b1000 | Default: 0000.0000

Bit	Reset	Description
3:0	0x8	ISPW_TMVAL

15.4.2.57 MC_TIMEOUT_MPCORE_0

Reset value defaults to 8 for most clients, but may be different for certain clients. If not zero, the client request time-out counter is reloaded with this value when it reaches zero. It is decremented when there is a pending request and the time-out clock counter for the request destination (internal/external memory) reaches zero.

When the client request time-out counter reaches zero, a number of time-out credits equal to the number of its requests pending inside the memory controller client FIFO are allocated to this client.

The number of time-out credits is decremented each time a request is granted. As long as there are credits, the client time-out priority bit is set.

Request time-out Register for CPU clients

Offset: 1d4h | Read/Write: R/W | Reset: 0b10001000 | Default: 0000.0000

Bit	Reset	Description
7:4	0x8	MPCOREW_TMVAL
3:0	0x8	MPCORER_TMVAL

15.4.2.58 MC_TIMEOUT_MPEA_0

Reset value defaults to 8 for most clients, but may be different for certain clients. If not zero, the client request time-out counter is reloaded with this value when it reaches zero. It is decremented when there is a pending request and the time-out clock counter for the request destination (internal/external memory) reaches zero.

When the client request time-out counter reaches zero, a number of time-out credits equal to the number of its requests pending inside the memory controller client FIFO are allocated to this client.

The number of time-out credits is decremented each time a request is granted. As long as there are credits, the client time-out priority bit is set.

Request time-out Register for mpea clients

Offset: 1d8h | Read/Write: R/W | Reset: 0b1000

Bit	Reset	Description
3:0	0x8	MPEAMEMRD_TMVAL

15.4.2.59 MC_TIMEOUT_MPEB_0

Reset value defaults to 8 for most clients, but may be different for certain clients. If not zero, the client request time-out counter is reloaded with this value when it reaches zero. It is decremented when there is a pending request and the time-out clock counter for the request destination (internal/external memory) reaches zero.

When the client request time-out counter reaches zero, a number of time-out credits equal to the number of its requests pending inside the memory controller client FIFO are allocated to this client.

The number of time-out credits is decremented each time a request is granted. As long as there are credits, the client time-out priority bit is set.

Request time-out Register for mpeb clients

Offset: 1dch | Read/Write: R/W | Reset: 0b100010001000

Bit	Reset	Description
11:8	0x8	MPEUNIFBW_TMVAL
7:4	0x8	MPE_IPRED_TMVAL
3:0	0x8	MPEUNIFBR_TMVAL

15.4.2.60 MC_TIMEOUT_MPEC_0

Reset value defaults to 8 for most clients, but may be different for certain clients. If not zero, the client request time-out counter is reloaded with this value when it reaches zero. It is decremented when there is a pending request and the time-out clock counter for the request destination (internal/external memory) reaches zero.

When the client request time-out counter reaches zero, a number of time-out credits equal to the number of its requests pending inside the memory controller client FIFO are allocated to this client.

The number of time-out credits is decremented each time a request is granted. As long as there are credits, the client time-out priority bit is set.

Request time-out Register for mpec clients

Offset: 1e0h | Read/Write: R/W | Reset: 0b10001000

Bit	Reset	Description
7:4	0x8	MPECSWR_TMVAL
3:0	0x8	MPECSRD_TMVAL

15.4.2.61 MC_TIMEOUT_NV_0

Reset value defaults to 8 for most clients, but may be different for certain clients. If not zero, the client request time-out counter is reloaded with this value when it reaches zero. It is decremented when there is a pending request and the time-out clock counter for the request destination (internal/external memory) reaches zero.

When the client request time-out counter reaches zero, a number of time-out credits equal to the number of its requests pending inside the memory controller client FIFO are allocated to this client.

The number of time-out credits is decremented each time a request is granted. As long as there are credits, the client time-out priority bit is set.

Request time-out Register for nv clients

Offset: 1e4h | Read/Write: R/W | Reset: 0b1000100010001000

Bit	Reset	Description
15:12	0x8	FDCDWR_TMVAL
11:8	0x8	TEXSRD_TMVAL
7:4	0x8	IDXSRD_TMVAL
3:0	0x8	FDCDRD_TMVAL

15.4.2.62 MC_TIMEOUT_PPCS_0

Reset value defaults to 8 for most clients, but may be different for certain clients. If not zero, the client request time-out counter is reloaded with this value when it reaches zero. It is decremented when there is a pending request and the time-out clock counter for the request destination (internal/external memory) reaches zero.

When the client request time-out counter reaches zero, a number of time-out credits equal to the number of its requests pending inside the memory controller client FIFO are allocated to this client.

The number of time-out credits is decremented each time a request is granted. As long as there are credits, the client time-out priority bit is set.

Request time-out Register for ppcs clients

Offset: 1e8h | Read/Write: R/W | Reset: 0b1000100010001000

Bit	Reset	Description
15:12	0x8	PPCSAHBSLVW_TMVAL
11:8	0x8	PPCSAHBDMAW_TMVAL
7:4	0x8	PPCSAHBSLVR_TMVAL
3:0	0x8	PPCSAHBDMAR_TMVAL

15.4.2.63 MC_TIMEOUT_VDE_0

Reset value defaults to 8 for most clients, but may be different for certain clients. If not zero, the client request time-out counter is reloaded with this value when it reaches zero. It is decremented when there is a pending request and the time-out clock counter for the request destination (internal/external memory) reaches zero.

When the client request time-out counter reaches zero, a number of time-out credits equal to the number of its requests pending inside the memory controller client FIFO are allocated to this client.

The number of time-out credits is decremented each time a request is granted. As long as there are credits, the client time-out priority bit is set.

Request time-out Register for vde clients

Offset: 1ech | Read/Write: R/W | Reset: 0b01000100010001000100010010

Bit	Reset	Description
27:24	0x4	VDETPMW_TMVAL
23:20	0x4	VDEMBEW_TMVAL
19:16	0x4	VDEBSEVW_TMVAL
15:12	0x4	VDETPER_TMVAL
11:8	0x4	VDEMCER_TMVAL
7:4	0x4	VDEMBER_TMVAL
3:0	0x4	VDEBSEVR_TMVAL

15.4.2.64 MC_TIMEOUT_VI_0

Reset value defaults to 8 for most clients, but may be different for certain clients. If not zero, the client request time-out counter is reloaded with this value when it reaches zero. It is decremented when there is a pending request and the time-out clock counter for the request destination (internal/external memory) reaches zero.

When the client request time-out counter reaches zero, a number of time-out credits equal to the number of its requests pending inside the memory controller client FIFO are allocated to this client.

The number of time-out credits is decremented each time a request is granted. As long as there are credits, the client time-out priority bit is set.

Request time-out Register for vi clients

Offset: 1f0h | Read/Write: R/W | Reset: 0b1000100010001000100

Bit	Reset	Description
19:16	0x8	VIWY_TMVAL
15:12	0x8	VIWV_TMVAL
11:8	0x8	VIWU_TMVAL
7:4	0x8	VIWSB_TMVAL
3:0	0x8	VIRUV_TMVAL

Read Coalescing Control Registers

Reset value defaults to 4 for most clients, but may be different for certain clients. Read coalescing happens inside the memory controller, right before arbitration. Coalescing means two (NV_MC_MW/2)-bit requests are grouped together in one NV_MC_WM-bit request.

The register value indicates how many cycles a first write request is going to wait for a subsequent one for possible coalescing. The coalescing can only happen if the request addresses are compatible. A value of zero means that coalescing is off and requests are sent right away to the arbiters.

Read coalescing can have a very significant impact performance when accessing the internal memory, because its memory word is NV_MC_WM-bit wide. Grouping two half-word accesses is much more efficient, because the two accesses would actually have taken three cycles, due to a stall when accessing the same memory bank. It also reduces the number of accessing (one instead of two), freeing up internal memory bandwidth for other accesses.

The impact on external memory accesses is not as significant as the burst access is for NV_MC_MW/2 bits. But a coalesced read guarantees two consecutive same page accesses which is good for external memory bandwidth utilization.

The read coalescing time-out should be programmed depending on the client behavior and the client versus memory controller clock ratio. The first read is obviously delayed by an amount of memory controller cycles equal to the time-out value.

15.4.2.65 MC_TIMEOUT_RCOAL_AVPC_0

Read Coalescing time-out register for AVPC read clients

Offset: 1f4h | Read/Write: R/W | Reset: 0b00000100

Bit	Reset	Description
7:0	0x4	AVPCARM7R_RCOAL_TMVAL

15.4.2.66 MC_TIMEOUT_RCOAL_DC_0

Reset value defaults to 4 for most clients, but may be different for certain clients. Read coalescing happens inside the memory controller, right before arbitration. Coalescing means two (NV_MC_MW/2)-bit requests are grouped together in one NV_MC_WM-bit request.

The register value indicates how many cycles a first write request is going to wait for a subsequent one for possible coalescing. The coalescing can only happen if the request addresses are compatible. A value of zero means that coalescing is off and requests are sent right away to the arbiters.

Read coalescing can have a very significant impact performance when accessing the internal memory, because its memory word is NV_MC_WM-bit wide. Grouping two half-word accesses is much more efficient, because the two accesses would actually have taken three cycles, due to a stall when accessing the same memory bank. It also reduces the number of accessing (one instead of two), freeing up internal memory bandwidth for other accesses.

The impact on external memory accesses is not as significant as the burst access is for NV_MC_MW/2 bits. But a coalesced read guarantees two consecutive same page accesses which is good for external memory bandwidth utilization.

The read coalescing time-out should be programmed depending on the client behavior and the client versus memory controller clock ratio. The first read is obviously delayed by an amount of memory controller cycles equal to the time-out value.

Read Coalescing time-out register for dc read clients

Offset: 1f8h | Read/Write: R/W | Reset: 0b00000100000001000000010000000100

Bit	Reset	Description
31:24	0x4	DISPLAY1B_RCOAL_TMVAL
23:16	0x4	DISPLAY0C_RCOAL_TMVAL
15:8	0x4	DISPLAY0B_RCOAL_TMVAL
7:0	0x4	DISPLAY0A_RCOAL_TMVAL

15.4.2.67 MC_TIMEOUT1_RCOAL_DC_0

Offset: 1fch | Read/Write: R/W | Reset: 0b00000100

Bit	Reset	Description
7:0	0x4	DISPLAYHC_RCOAL_TMVAL

15.4.2.68 MC_TIMEOUT_RCOAL_DCB_0

Reset value defaults to 4 for most clients, but may be different for certain clients. Read coalescing happens inside the memory controller, right before arbitration. Coalescing means two (NV_MC_MW/2)-bit requests are grouped together in one NV_MC_WM-bit request.

The register value indicates how many cycles a first write request is going to wait for a subsequent one for possible coalescing. The coalescing can only happen if the request addresses are compatible. A value of zero means that coalescing is off and requests are sent right away to the arbiters.

Read coalescing can have a very significant impact performance when accessing the internal memory, because its memory word is NV_MC_WM-bit wide. Grouping two half-word accesses is much more efficient, because the two accesses would actually have taken three cycles, due to a stall when accessing the same memory bank. It also reduces the number of accessing (one instead of two), freeing up internal memory bandwidth for other accesses.

The impact on external memory accesses is not as significant as the burst access is for NV_MC_MW/2 bits. But a coalesced read guarantees two consecutive same page accesses which is good for external memory bandwidth utilization.

The read coalescing time-out should be programmed depending on the client behavior and the client versus memory controller clock ratio. The first read is obviously delayed by an amount of memory controller cycles equal to the time-out value.

Read Coalescing time-out register for dcb read clients

Offset: 200h | Read/Write: R/W | Reset: 0b00000100000001000000010000000100

Bit	Reset	Description
31:24	0x4	DISPLAY1BB_RCOAL_TMVAL
23:16	0x4	DISPLAY0CB_RCOAL_TMVAL

Bit	Reset	Description
15:8	0x4	DISPLAY0BB_RCOAL_TMVAL
7:0	0x4	DISPLAY0AB_RCOAL_TMVAL

15.4.2.69 MC_TIMEOUT1_RCOAL_DCB_0

Offset: 204h | Read/Write: R/W | Reset: 0b00000100

Bit	Reset	Description
7:0	0x4	DISPLAYHCB_RCOAL_TMVAL

15.4.2.70 MC_TIMEOUT_RCOAL_EPP_0

Reset value defaults to 4 for most clients, but may be different for certain clients. Read coalescing happens inside the memory controller, right before arbitration. Coalescing means two (NV_MC_MW/2)-bit requests are grouped together in one NV_MC_WM-bit request.

The register value indicates how many cycles a first write request is going to wait for a subsequent one for possible coalescing. The coalescing can only happen if the request addresses are compatible. A value of zero means that coalescing is off and requests are sent right away to the arbiters.

Read coalescing can have a very significant impact performance when accessing the internal memory, because its memory word is NV_MC_WM-bit wide. Grouping two half-word accesses is much more efficient, because the two accesses would actually have taken three cycles, due to a stall when accessing the same memory bank. It also reduces the number of accessing (one instead of two), freeing up internal memory bandwidth for other accesses.

The impact on external memory accesses is not as significant as the burst access is for NV_MC_MW/2 bits. But a coalesced read guarantees two consecutive same page accesses which is good for external memory bandwidth utilization.

The read coalescing time-out should be programmed depending on the client behavior and the client versus memory controller clock ratio. The first read is obviously delayed by an amount of memory controller cycles equal to the time-out value.

Read Coalescing time-out register for epp read clients

Offset: 208h | Read/Write: R/W | Reset: 0b00000100

Bit	Reset	Description
7:0	0x4	EPPUP_RCOAL_TMVAL

15.4.2.71 MC_TIMEOUT_RCOAL_G2_0

Reset value defaults to 4 for most clients, but may be different for certain clients. Read coalescing happens inside the memory controller, right before arbitration. Coalescing means two (NV_MC_MW/2)-bit requests are grouped together in one NV_MC_WM-bit request.

The register value indicates how many cycles a first write request is going to wait for a subsequent one for possible coalescing. The coalescing can only happen if the request addresses are compatible. A value of zero means that coalescing is off and requests are sent right away to the arbiters.

Read coalescing can have a very significant impact performance when accessing the internal memory, because its memory word is NV_MC_WM-bit wide. Grouping two half-word accesses is much more efficient, because the two accesses would actually have taken three cycles, due to a stall when accessing the same memory bank. It also reduces the number of accessing (one instead of two), freeing up internal memory bandwidth for other accesses.

The impact on external memory accesses is not as significant as the burst access is for NV_MC_MW/2 bits. But a coalesced read guarantees two consecutive same page accesses which is good for external memory bandwidth utilization.

The read coalescing time-out should be programmed depending on the client behavior and the client versus memory controller clock ratio. The first read is obviously delayed by an amount of memory controller cycles equal to the time-out value.

Read Coalescing time-out register for g2 read clients

Offset: 20ch | Read/Write: R/W | Reset: 0b00000100000010000000100

Bit	Reset	Description
23:16	0x4	G2DR_RCOAL_TMVAL
15:8	0x4	G2SR_RCOAL_TMVAL
7:0	0x4	G2PR_RCOAL_TMVAL

15.4.2.72 MC_TIMEOUT_RCOAL_HC_0

Reset value defaults to 4 for most clients, but may be different for certain clients. Read coalescing happens inside the memory controller, right before arbitration. Coalescing means two (NV_MC_MW/2)-bit requests are grouped together in one NV_MC_WM-bit request.

The register value indicates how many cycles a first write request is going to wait for a subsequent one for possible coalescing. The coalescing can only happen if the request addresses are compatible. A value of zero means that coalescing is off and requests are sent right away to the arbiters.

Read coalescing can have a very significant impact performance when accessing the internal memory, because its memory word is NV_MC_WM-bit wide. Grouping two half-word accesses is much more efficient, because the two accesses would actually have taken three cycles, due to a stall when accessing the same memory bank. It also reduces the number of accessing (one instead of two), freeing up internal memory bandwidth for other accesses.

The impact on external memory accesses is not as significant as the burst access is for NV_MC_MW/2 bits. But a coalesced read guarantees two consecutive same page accesses which is good for external memory bandwidth utilization.

The read coalescing time-out should be programmed depending on the client behavior and the client versus memory controller clock ratio. The first read is obviously delayed by an amount of memory controller cycles equal to the time-out value.

Read Coalescing time-out register for hc read clients

Offset: 210h | Read/Write: R/W | Reset: 0b0000010000000100

Bit	Reset	Description
15:8	0x4	HOST1XR_RCOAL_TMVAL
7:0	0x4	HOST1XDMAR_RCOAL_TMVAL

15.4.2.73 MC_TIMEOUT_RCOAL_MPCORE_0

Reset value defaults to 4 for most clients, but may be different for certain clients. Read coalescing happens inside the memory controller, right before arbitration. Coalescing means two (NV_MC_MW/2)-bit requests are grouped together in one NV_MC_WM-bit request.

The register value indicates how many cycles a first write request is going to wait for a subsequent one for possible coalescing. The coalescing can only happen if the request addresses are compatible. A value of zero means that coalescing is off and requests are sent right away to the arbiters.

Read coalescing can have a very significant impact performance when accessing the internal memory, because its memory word is NV_MC_WM-bit wide. Grouping two half-word accesses is much more efficient, because the two accesses would actually have taken three cycles, due to a stall when accessing the same memory bank. It also reduces the number of accessing (one instead of two), freeing up internal memory bandwidth for other accesses.

The impact on external memory accesses is not as significant as the burst access is for NV_MC_MW/2 bits. But a coalesced read guarantees two consecutive same page accesses which is good for external memory bandwidth utilization.

The read coalescing time-out should be programmed depending on the client behavior and the client versus memory controller clock ratio. The first read is obviously delayed by an amount of memory controller cycles equal to the time-out value.

Read Coalescing time-out register for CPU read clients

Offset: 214h | Read/Write: R/W | Reset: 0b00000100

Bit	Reset	Description
7:0	0x4	MPCORER_RCOAL_TMVAL

15.4.2.74 MC_TIMEOUT_RCOAL_MPEA_0

Reset value defaults to 4 for most clients, but may be different for certain clients. Read coalescing happens inside the memory controller, right before arbitration. Coalescing means two (NV_MC_MW/2)-bit requests are grouped together in one NV_MC_WM-bit request.

The register value indicates how many cycles a first write request is going to wait for a subsequent one for possible coalescing. The coalescing can only happen if the request addresses are compatible. A value of zero means that coalescing is off and requests are sent right away to the arbiters.

Read coalescing can have a very significant impact performance when accessing the internal memory, because its memory word is NV_MC_WM-bit wide. Grouping two half-word accesses is much more efficient, because the two accesses would actually have taken three cycles, due to a stall when accessing the same memory bank. It also reduces the number of accessing (one instead of two), freeing up internal memory bandwidth for other accesses.

The impact on external memory accesses is not as significant as the burst access is for NV_MC_MW/2 bits. But a coalesced read guarantees two consecutive same page accesses which is good for external memory bandwidth utilization.

The read coalescing time-out should be programmed depending on the client behavior and the client versus memory controller clock ratio. The first read is obviously delayed by an amount of memory controller cycles equal to the time-out value.

Read Coalescing time-out register for mpea read clients

Offset: 218h | Read/Write: R/W | Reset: 0b00000100

Bit	Reset	Description
7:0	0x4	MPEAMEMRD_RCOAL_TMVAL

15.4.2.75 MC_TIMEOUT_RCOAL_MPEB_0

Reset value defaults to 4 for most clients, but may be different for certain clients. Read coalescing happens inside the memory controller, right before arbitration. Coalescing means two (NV_MC_MW/2)-bit requests are grouped together in one NV_MC_WM-bit request.

The register value indicates how many cycles a first write request is going to wait for a subsequent one for possible coalescing. The coalescing can only happen if the request addresses are compatible. A value of zero means that coalescing is off and requests are sent right away to the arbiters.

Read coalescing can have a very significant impact performance when accessing the internal memory, because its memory word is NV_MC_WM-bit wide. Grouping two half-word accesses is much more efficient, because the two accesses would actually have taken three cycles, due to a stall when accessing the same memory bank. It also reduces the number of accessing (one instead of two), freeing up internal memory bandwidth for other accesses.

The impact on external memory accesses is not as significant as the burst access is for NV_MC_MW/2 bits. But a coalesced read guarantees two consecutive same page accesses which is good for external memory bandwidth utilization.

The read coalescing time-out should be programmed depending on the client behavior and the client versus memory controller clock ratio. The first read is obviously delayed by an amount of memory controller cycles equal to the time-out value.

Read Coalescing time-out register for mpeb read clients

Offset: 21ch | Read/Write: R/W | Reset: 0b0000010000000100

Bit	Reset	Description
15:8	0x4	MPE_IPRED_RCOAL_TMVAL
7:0	0x4	MPEUNIFBR_RCOAL_TMVAL

15.4.2.76 MC_TIMEOUT_RCOAL_MPEC_0

Reset value defaults to 4 for most clients, but may be different for certain clients. Read coalescing happens inside the memory controller, right before arbitration. Coalescing means two (NV_MC_MW/2)-bit requests are grouped together in one NV_MC_WM-bit request.

The register value indicates how many cycles a first write request is going to wait for a subsequent one for possible coalescing. The coalescing can only happen if the request addresses are compatible. A value of zero means that coalescing is off and requests are sent right away to the arbiters.

Read coalescing can have a very significant impact performance when accessing the internal memory, because its memory word is NV_MC_WM-bit wide. Grouping two half-word accesses is much more efficient, because the two accesses would actually have taken three cycles, due to a stall when accessing the same memory bank. It also reduces the number of accessing (one instead of two), freeing up internal memory bandwidth for other accesses.

The impact on external memory accesses is not as significant as the burst access is for NV_MC_MW/2 bits. But a coalesced read guarantees two consecutive same page accesses which is good for external memory bandwidth utilization.

The read coalescing time-out should be programmed depending on the client behavior and the client versus memory controller clock ratio. The first read is obviously delayed by an amount of memory controller cycles equal to the time-out value.

Read Coalescing time-out register for mpec read clients

Offset: 220h | Read/Write: R/W | Reset: 0b00000100

Bit	Reset	Description
7:0	0x4	MPECSRD_RCOAL_TMVAL

15.4.2.77 MC_TIMEOUT_RCOAL_NV_0

Reset value defaults to 4 for most clients, but may be different for certain clients. Read coalescing happens inside the memory controller, right before arbitration. Coalescing means two (NV_MC_MW/2)-bit requests are grouped together in one NV_MC_WM-bit request.

The register value indicates how many cycles a first write request is going to wait for a subsequent one for possible coalescing. The coalescing can only happen if the request addresses are compatible. A value of zero means that coalescing is off and requests are sent right away to the arbiters.

Read coalescing can have a very significant impact performance when accessing the internal memory, because its memory word is NV_MC_WM-bit wide. Grouping two half-word accesses is much more efficient, because the two accesses would actually have taken three cycles, due to a stall when accessing the same memory bank. It also reduces the number of accessing (one instead of two), freeing up internal memory bandwidth for other accesses.

The impact on external memory accesses is not as significant as the burst access is for NV_MC_MW/2 bits. But a coalesced read guarantees two consecutive same page accesses which is good for external memory bandwidth utilization.

The read coalescing time-out should be programmed depending on the client behavior and the client versus memory controller clock ratio. The first read is obviously delayed by an amount of memory controller cycles equal to the time-out value.

Read Coalescing time-out register for nv read clients

Offset: 224h | Read/Write: R/W | Reset: 0b0000010000000100

Bit	Reset	Description
15:8	0x4	TEXSRD_RCOAL_TMVAL
7:0	0x4	IDXSRD_RCOAL_TMVAL

15.4.2.78 MC_TIMEOUT_RCOAL_PPCS_0

Reset value defaults to 4 for most clients, but may be different for certain clients. Read coalescing happens inside the memory controller, right before arbitration. Coalescing means two (NV_MC_MW/2)-bit requests are grouped together in one NV_MC_WM-bit request.

The register value indicates how many cycles a first write request is going to wait for a subsequent one for possible coalescing. The coalescing can only happen if the request addresses are compatible. A value of zero means that coalescing is off and requests are sent right away to the arbiters.

Read coalescing can have a very significant impact performance when accessing the internal memory, because its memory word is NV_MC_WM-bit wide. Grouping two half-word accesses is much more efficient, because the two accesses would actually have taken three cycles, due to a stall when accessing the same memory bank. It also reduces the number of accessing (one instead of two), freeing up internal memory bandwidth for other accesses.

The impact on external memory accesses is not as significant as the burst access is for NV_MC_MW/2 bits. But a coalesced read guarantees two consecutive same page accesses which is good for external memory bandwidth utilization.

The read coalescing time-out should be programmed depending on the client behavior and the client versus memory controller clock ratio. The first read is obviously delayed by an amount of memory controller cycles equal to the time-out value.

Read Coalescing time-out register for ppcs read clients

Offset: 228h | Read/Write: R/W | Reset: 0b0000010000000100

Bit	Reset	Description
15:8	0x4	PPCSAHBSLVR_RCOAL_TMVAL
7:0	0x4	PPCSAHBDMAR_RCOAL_TMVAL

15.4.2.79 MC_TIMEOUT_RCOAL_VDE_0

Reset value defaults to 4 for most clients, but may be different for certain clients. Read coalescing happens inside the memory controller, right before arbitration. Coalescing means two (NV_MC_MW/2)-bit requests are grouped together in one NV_MC_WM-bit request.

The register value indicates how many cycles a first write request is going to wait for a subsequent one for possible coalescing. The coalescing can only happen if the request addresses are compatible. A value of zero means that coalescing is off and requests are sent right away to the arbiters.

Read coalescing can have a very significant impact performance when accessing the internal memory, because its memory word is NV_MC_WM-bit wide. Grouping two half-word accesses is much more efficient, because the two accesses would actually have taken three cycles, due to a stall when accessing the same memory bank. It also reduces the number of accessing (one instead of two), freeing up internal memory bandwidth for other accesses.

The impact on external memory accesses is not as significant as the burst access is for NV_MC_MW/2 bits. But a coalesced read guarantees two consecutive same page accesses which is good for external memory bandwidth utilization.

The read coalescing time-out should be programmed depending on the client behavior and the client versus memory controller clock ratio. The first read is obviously delayed by an amount of memory controller cycles equal to the time-out value.

Read Coalescing time-out register for vde read clients

Offset: 22ch | Read/Write: R/W | Reset: 0b00000100000001000000010000000100

Bit	Reset	Description
31:24	0x4	VDETPER_RCOAL_TMVAL
23:16	0x4	VDEMCER_RCOAL_TMVAL
15:8	0x4	VDEMBER_RCOAL_TMVAL
7:0	0x4	VDEBSEVR_RCOAL_TMVAL

15.4.2.80 MC_TIMEOUT_RCOAL_VI_0

Reset value defaults to 4 for most clients, but may be different for certain clients. Read coalescing happens inside the memory controller, right before arbitration. Coalescing means two (NV_MC_MW/2)-bit requests are grouped together in one NV_MC_WM-bit request.

The register value indicates how many cycles a first write request is going to wait for a subsequent one for possible coalescing. The coalescing can only happen if the request addresses are compatible. A value of zero means that coalescing is off and requests are sent right away to the arbiters.

Read coalescing can have a very significant impact performance when accessing the internal memory, because its memory word is NV_MC_WM-bit wide. Grouping two half-word accesses is much more efficient, because the two accesses would actually have taken three cycles, due to a stall when accessing the same memory bank. It also reduces the number of accessing (one instead of two), freeing up internal memory bandwidth for other accesses.

The impact on external memory accesses is not as significant as the burst access is for NV_MC_MW/2 bits. But a coalesced read guarantees two consecutive same page accesses which is good for external memory bandwidth utilization.

The read coalescing time-out should be programmed depending on the client behavior and the client versus memory controller clock ratio. The first read is obviously delayed by an amount of memory controller cycles equal to the time-out value.

Read Coalescing time-out register for vi read clients

Offset: 230h | Read/Write: R/W | Reset: 0b00000100

Bit	Reset	Description
7:0	0x4	VIRUV_RCOAL_TMVAL

15.4.2.81 MC_RCOAL_AUTODISABLE_0_0

Reset value is enabled. Control whether read coalesce logic for the listed block-read clients will attempt to detect when the client is producing a long sequence of non-coalescable requests (as is common in horizontal-linear walks through a tiled surface) and proactively disable read-coalescing while the client appears to be in that mode.

Read Coalescing Autodisable Control Register.

Offset: 234h | Read/Write: R/W | Reset: 0b1111111111111

Bit	Reset	Description
12	ENABLE	VIRUV_RCOAL_AUTODISABLE_EN: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
11	ENABLE	MPEUNIFBR_RCOAL_AUTODISABLE_EN: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
10	ENABLE	G2SR_RCOAL_AUTODISABLE_EN: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
9	ENABLE	G2PR_RCOAL_AUTODISABLE_EN: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
8	ENABLE	EPPUP_RCOAL_AUTODISABLE_EN: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
7	ENABLE	DISPLAY1BB_RCOAL_AUTODISABLE_EN: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
6	ENABLE	DISPLAY1B_RCOAL_AUTODISABLE_EN: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
5	ENABLE	DISPLAY0CB_RCOAL_AUTODISABLE_EN: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED

Bit	Reset	Description
4	ENABLE	DISPLAY0C_RCOAL_AUTODISABLE_EN: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
3	ENABLE	DISPLAY0BB_RCOAL_AUTODISABLE_EN: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
2	ENABLE	DISPLAY0B_RCOAL_AUTODISABLE_EN: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
1	ENABLE	DISPLAY0AB_RCOAL_AUTODISABLE_EN: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
0	ENABLE	DISPLAY0A_RCOAL_AUTODISABLE_EN: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED

15.4.2.82 MC_BWSHARE_AVPC_0

Reset value is 0. The values reflect the bandwidth share for EMEM.

INC_VAL specifies the number of bytes the Bandwidth credit has to be incremented. A value of zero disables the bandwidth share for all AVPC clients, and reset the accumulated credit.

HIGH_TH specifies the high threshold [11:4]. If accumulated credit >= high threshold, BW bit is set. BW bit is reset when accumulated credit is zero. SAT_TH specifies the saturation threshold [11:4] for accumulated credit.

Bandwidth share register for AVPC clients

Offset: 238h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
28	TM_SFACTOR1	AVPC_BW_TMSFACTORSEL: 0 = TM_SFACTOR1 1 = TM_SFACTOR2
27	DISABLE	AVPC_BW_ALWAYSINC: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
26:19	0x0	AVPC_BW_MAXTH
18:11	0x0	AVPC_BW_HIGHTH
10:0	0x0	AVPC_BW_INCVAL

15.4.2.83 MC_BWSHARE_DC_0

Reset value is 0. The values reflect the bandwidth share for EMEM.

INC_VAL specifies the number of bytes the Bandwidth credit has to be incremented. A value of zero disables the bandwidth share for all DC clients, and reset the accumulated credit.

HIGH_TH specifies the high threshold [11:4]. If accumulated credit >= high threshold, BW bit is set. BW bit is reset when accumulated credit is zero. SAT_TH specifies the saturation threshold [11:4] for accumulated credit.

Bandwidth share register for DC clients

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

Bit	Reset	Description
28	TM_SFACTOR1	DC_BW_TMSFACTORSEL: 0 = TM_SFACTOR1 1 = TM_SFACTOR2
27	DISABLE	DC_BW_ALWAYSINC: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
26:19	0x0	DC_BW_MAXTH
18:11	0x0	DC_BW_HIGHTH
10:0	0x0	DC_BW_INCVAL

15.4.2.84 MC_BWSHARE_DCB_0

Reset value is 0. The values reflect the bandwidth share for EMEM.

INC_VAL specifies the number of bytes the Bandwidth credit has to be incremented. A value of zero disables the bandwidth share for all DCB clients, and reset the accumulated credit.

HIGH_TH specifies the high threshold [11:4]. If accumulated credit >= high threshold, BW bit is set. BW bit is reset when accumulated credit is zero. SAT_TH specifies the saturation threshold [11:4] for accumulated credit.

Bandwidth share register for DCB clients

Offset: 240h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
28	TM_SFACTOR1	DCB_BW_TMSFACTORSEL: 0 = TM_SFACTOR1 1 = TM_SFACTOR2
27	DISABLE	DCB_BW_ALWAYSINC: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
26:19	0x0	DCB_BW_MAXTH
18:11	0x0	DCB_BW_HIGHTH
10:0	0x0	DCB_BW_INCVAL

15.4.2.85 MC_BWSHARE_EPP_0

The values reflect the bandwidth share for EMEM. Reset value is 0.

INC_VAL specifies the number of bytes the Bandwidth credit has to be incremented. A value of zero disables the bandwidth share for all EPP clients, and reset the accumulated credit.

HIGH_TH specifies the high threshold[11:4]. If accumulated credit >= high threshold, BW bit is set. BW bit is reset when accumulated credit is zero. SAT_TH specifies the saturation threshold[11:4] for accumulated credit.

Bandwidth share register for EPP clients

Offset: 244h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
28	TM_SFACTOR1	EPP_BW_TMSFACTORSEL: 0 = TM_SFACTOR1 1 = TM_SFACTOR2
27	DISABLE	EPP_BW_ALWAYSINC: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
26:19	0x0	EPP_BW_MAXTH
18:11	0x0	EPP_BW_HIGHTH
10:0	0x0	EPP_BW_INCVAL

15.4.2.86 MC_BWSHARE_G2_0

The values reflect the bandwidth share for EMEM. Reset value is 0.

INC_VAL specifies the number of bytes the Bandwidth credit has to be incremented. A value of zero disables the bandwidth share for all G2 clients, and reset the accumulated credit.

HIGH_TH specifies the high threshold[11:4]. If accumulated credit >= high threshold, BW bit is set. BW bit is reset when accumulated credit is zero. SAT_TH specifies the saturation threshold[11:4] for accumulated credit.

Bandwidth share register for G2 clients

Offset: 248h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
28	TM_SFACTOR1	G2_BW_TMSFACTORSEL: 0 = TM_SFACTOR1 1 = TM_SFACTOR2
27	DISABLE	G2_BW_ALWAYSINC: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
26:19	0x0	G2_BW_MAXTH
18:11	0x0	G2_BW_HIGHTH

Bit	Reset	Description
10:0	0x0	G2_BW_INCVAL

15.4.2.87 MC_BWSHARE_HC_0

The values reflect the bandwidth share for EMEM. Reset value is 0.

INC_VAL specifies the number of bytes the Bandwidth credit has to be incremented. A value of zero disables the bandwidth share for all HC clients, and reset the accumulated credit.

HIGH_TH specifies the high threshold[11:4]. If accumulated credit >= high threshold, BW bit is set. BW bit is reset when accumulated credit is zero. SAT_TH specifies the saturation threshold[11:4] for accumulated credit.

Bandwidth share register for HC clients

Offset: 24ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
28	TM_SFACTOR1	HC_BW_TMSFACTORSEL: 0 = TM_SFACTOR1 1 = TM_SFACTOR2
27	DISABLE	HC_BW_ALWAYSINC: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
26:19	0x0	HC_BW_MAXTH
18:11	0x0	HC_BW_HIGHTH
10:0	0x0	HC_BW_INCVAL

15.4.2.88 MC_BWSHARE_ISP_0

The values reflect the bandwidth share for EMEM. Reset value is 0.

INC_VAL specifies the number of bytes the Bandwidth credit has to be incremented. A value of zero disables the bandwidth share for all ISP clients, and reset the accumulated credit.

HIGH_TH specifies the high threshold[11:4]. If accumulated credit >= high threshold, BW bit is set. BW bit is reset when accumulated credit is zero. SAT_TH specifies the saturation threshold[11:4] for accumulated credit.

Bandwidth share register for ISP clients

Offset: 250h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
28	TM_SFACTOR1	ISP_BW_TMSFACTORSEL: 0 = TM_SFACTOR1 1 = TM_SFACTOR2
27	DISABLE	ISP_BW_ALWAYSINC: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
26:19	0x0	ISP_BW_MAXTH
18:11	0x0	ISP_BW_HIGHTH
10:0	0x0	ISP_BW_INCVAL

15.4.2.89 MC_BWSHARE_MPCORE_0

The values reflect the bandwidth share for EMEM. Reset value is 0.

INC_VAL specifies the number of bytes the Bandwidth credit has to be incremented. A value of zero disables the bandwidth share for all CPU clients, and reset the accumulated credit.

HIGH_TH specifies the high threshold[11:4]. If accumulated credit >= high threshold, BW bit is set. BW bit is reset when accumulated credit is zero. SAT_TH specifies the saturation threshold[11:4] for accumulated credit.

Bandwidth share register for CPU clients

Offset: 254h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
28	TM_SFACTOR1	MPCORE_BW_TMSFACTORSEL: 0 = TM_SFACTOR1 1 = TM_SFACTOR2
27	DISABLE	MPCORE_BW_ALWAYSINC: 0 = DISABLE 1 = ENABLE
26:19	0x0	MPCORE_BW_MAXTH
18:11	0x0	MPCORE_BW_HIGHTH
10:0	0x0	MPCORE_BW_INCVAL

15.4.2.90 MC_BWSHARE_MPEA_0

The values reflect the bandwidth share for EMEM. Reset value is 0.

INC_VAL specifies the number of bytes the Bandwidth credit has to be incremented. A value of zero disables the bandwidth share for all MPEA clients, and reset the accumulated credit.

HIGH_TH specifies the high threshold[11:4]. If accumulated credit >= high threshold, BW bit is set. BW bit is reset when accumulated credit is zero. SAT_TH specifies the saturation threshold[11:4] for accumulated credit.

Bandwidth share register for MPEA clients

Offset: 258h| Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
28	TM_SFACTOR1	MPEA_BW_TMSFACTORSEL: 0 = TM_SFACTOR1 1 = TM_SFACTOR2

Bit	Reset	Description
27	DISABLE	MPEA_BW_ALWAYSINC: 0 = DISABLE 1 = ENABLE
26:19	0x0	MPEA_BW_MAXTH
18:11	0x0	MPEA_BW_HIGHTH
10:0	0x0	MPEA_BW_INCVAL

15.4.2.91 MC_BWSHARE_MPEB_0

The values reflect the bandwidth share for EMEM. Reset value is 0.

INC_VAL specifies the number of bytes the Bandwidth credit has to be incremented. A value of zero disables the bandwidth share for all MPEB clients, and reset the accumulated credit.

HIGH_TH specifies the high threshold[11:4]. If accumulated credit >= high threshold, BW bit is set. BW bit is reset when accumulated credit is zero. SAT_TH specifies the saturation threshold[11:4] for accumulated credit.

Bandwidth share register for MPEB clients

Offset: 25ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
28	TM_SFACTOR1	MPEB_BW_TMSFACTORSEL: 0 = TM_SFACTOR1 1 = TM_SFACTOR2
27	DISABLE	MPEB_BW_ALWAYSINC: 0 = DISABLE 1 = ENABLE
26:19	0x0	MPEB_BW_MAXTH
18:11	0x0	MPEB_BW_HIGHTH
10:0	0x0	MPEB_BW_INCVAL

15.4.2.92 MC_BWSHARE_MPEC_0

The values reflect the bandwidth share for EMEM. Reset value is 0.

INC_VAL specifies the number of bytes the Bandwidth credit has to be incremented. A value of zero disables the bandwidth share for all MPEC clients, and reset the accumulated credit.

HIGH_TH specifies the high threshold[11:4]. If accumulated credit >= high threshold, BW bit is set. BW bit is reset when accumulated credit is zero. SAT_TH specifies the saturation threshold[11:4] for accumulated credit.

Bandwidth share register for MPEC clients

Offset: 260h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description

Bit	Reset	Description
28	TM_SFACTOR1	MPEC_BW_TMSFACTORSEL: 0 = TM_SFACTOR1 1 = TM_SFACTOR2
27	DISABLE	MPEC_BW_ALWAYSINC: 0 = DISABLE 1 = ENABLE
26:19	0x0	MPEC_BW_MAXTH
18:11	0x0	MPEC_BW_HIGHTH
10:0	0x0	MPEC_BW_INCVAL

15.4.2.93 MC_BWSHARE_NV_0

The values reflect the bandwidth share for EMEM. Reset value is 0.

INC_VAL specifies the number of bytes the Bandwidth credit has to be incremented. A value of zero disables the bandwidth share for all NV clients, and reset the accumulated credit.

HIGH_TH specifies the high threshold[11:4]. If accumulated credit >= high threshold, BW bit is set. BW bit is reset when accumulated credit is zero. SAT_TH specifies the saturation threshold[11:4] for accumulated credit.

Bandwidth share register for NV clients

Offset: 264h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
28	TM_SFACTOR1	NV_BW_TMSFACTORSEL: 0 = TM_SFACTOR1 1 = TM_SFACTOR2
27	DISABLE	NV_BW_ALWAYSINC: 0 = DISABLE 1 = ENABLE
26:19	0x0	NV_BW_MAXTH
18:11	0x0	NV_BW_HIGHTH
10:0	0x0	NV_BW_INCVAL

15.4.2.94 MC_BWSHARE_PPCS_0

The values reflect the bandwidth share for EMEM. Reset value is 0.

INC_VAL specifies the number of bytes the Bandwidth credit has to be incremented. A value of zero disables the bandwidth share for all PPCS clients, and reset the accumulated credit.

HIGH_TH specifies the high threshold[11:4]. If accumulated credit >= high threshold, BW bit is set. BW bit is reset when accumulated credit is zero. SAT_TH specifies the saturation threshold[11:4] for accumulated credit.

Bandwidth share register for PPCS clients

Offset: 268h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
28	TM_SFACTOR1	PPCS_BW_TMSFACTORSEL: 0 = TM_SFACTOR1 1 = TM_SFACTOR2
27	DISABLE	PPCS_BW_ALWAYSINC: 0 = DISABLE 1 = ENABLE
26:19	0x0	PPCS_BW_MAXTH
18:11	0x0	PPCS_BW_HIGHTH
10:0	0x0	PPCS_BW_INCVAL

15.4.2.95 MC_BWSHARE_VDE_0

The values reflect the bandwidth share for EMEM. Reset value is 0.

INC_VAL specifies the number of bytes the Bandwidth credit has to be incremented. A value of zero disables the bandwidth share for all VDE clients, and reset the accumulated credit.

HIGH_TH specifies the high threshold[11:4]. If accumulated credit >= high threshold, BW bit is set. BW bit is reset when accumulated credit is zero. SAT_TH specifies the saturation threshold[11:4] for accumulated credit.

Bandwidth share register for VDE clients

Offset: 26ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
28	TM_SFACTOR1	VDE_BW_TMSFACTORSEL: 0 = TM_SFACTOR1 1 = TM_SFACTOR2
27	DISABLE	VDE_BW_ALWAYSINC: 0 = DISABLE 1 = ENABLE
26:19	0x0	VDE_BW_MAXTH
18:11	0x0	VDE_BW_HIGHTH
10:0	0x0	VDE_BW_INCVAL

15.4.2.96 MC_BWSHARE_VI_0

The values reflect the bandwidth share for EMEM. Reset value is 0.

INC_VAL specifies the number of bytes the Bandwidth credit has to be incremented. A value of zero disables the bandwidth share for all VI clients, and reset the accumulated credit.

HIGH_TH specifies the high threshold[11:4]. If accumulated credit >= high threshold, BW bit is set. BW bit is reset when accumulated credit is zero. SAT_TH specifies the saturation threshold[11:4] for accumulated credit.

Bandwidth share register for VI clients

Offset: 270h| Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
28	TM_SFACTOR1	VI_BW_TMSFACTORSEL: 0 = TM_SFACTOR1 1 = TM_SFACTOR2
27	DISABLE	VI_BW_ALWAYSINC: 0 = DISABLE 1 = ENABLE
26:19	0x0	VI_BW_MAXTH
18:11	0x0	VI_BW_HIGHTH
10:0	0x0	VI_BW_INCVAL

16.0 NAND FLASH CONTROLLER

The NAND flash controller allows Tegra® 2 Processor to access NAND flash memories for mass storage. It supports both asynchronous (legacy) interfaces, and the ONFI 1.0 standard with error protection scheme required for data integrity in SLC and MLC devices. Up to 8 chip selects are supported natively, with more possible through GPIOs. The NAND flash controller is an AHB master and can initiate high speed data transfers between external DRAM and NAND flash memory through hardware buffering. NAND controller registers are connected on the APB interface as a slave, for register read/write access.

16.1 Features

The NAND Flash controller supports both PIO mode and DMA mode of operations for NAND flash access. Interrupts are generated upon a COMMAND sequence completion, as programmed in the NAND_COMMAND register. Supported operations include individual cycles of command, address, read data and write data; or a combined sequence for program, erase and read operations. When hardware ECC is enabled, data transfer cycles are extended to update the flash spare area to store the parity data information.

In PIO transfer modes, data written into the RESP register is transmitted out, and similarly read data is captured in the RESP register. Typically READ ID or READ STATUS sequences should make use of PIO mode of operation.

In DMA mode, write data from DRAM is transmitted out, while for read transfers, data received from flash is transferred to DRAM as programmed in the DMA configuration registers. For each of the data pipelines, a separate FIFO is used to throttle the Read/Write transfers.

The NAND controller supports HW_ERROR_CORRECTION without using large buffers for page size data storage in the ECC decoder/encoder.

The automatic RBSY status mode of operation reduces the software overhead, by checking the RBSY line status for CMB_RBSY application.

The NAND controller supports Command queue mode of operation targeted to reduce the software overhead in managing the IO transfers on page basis. A number of Flash operations can be queued in this mode of operation with NAND DMA interface that will bring in these commands and execute them in sequence.

16.1.1 NAND Controller Feature Summary

- Supports asynchronous interface requirements of various NAND Flash memory vendors.
- Programmable cycles to support multiple commands/operations in single sequence.
- Programmable timing interface.
- 8/16 bit data interface support.
- Optional Reed-Solomon/Hamming/BCH coding based ECC/EDC for main area data.
- Hamming/BCH based ECC/EDC for spare area data.
- Programmable error correction capability of t=4,6,8 symbol errors for each 512 bytes of data for Reed-Solomon selection of main area data.
- Programmable error correction capability of t=4,8,12, and 14 bit error for each 512 bytes of data for BCH selection of main and spare area.
- Programmable page size ranging from 256 bytes to 4K bytes.
- EDO mode of read access for maximum read throughput.
- Automated hardware status check.
- Command queue mode of operation to queue up a number of flash operations.

- Interface support up to 8 different devices.
- Single DMA based programming interface for multi-page read/write transfer.
- Software control to interleave operations on multiple cards for optimum performance.
- Extended decode status information to support software wear leveling and bad block management.

16.1.2 Software Features

The following is the list of Flash commands supported.

- Flash Program (write)
- Flash Read
- Single plane Cache Write
- Double plane Cache Write
- Flash Cache Read
- Block Erase
- Read Status
- Read Status Multi-plane
- Reset
- Random Data Input/Output

16.1.3 Hardware / Software Partitioning

Error detection is done in hardware and error correction is possible either in s/w or hardware. In this context, s/w error correction means, hardware will provide the decode results in the form of vectors consisting of error location, bits to be flipped to correct error and which region of flash this error has occurred. Software has to manage these error vector information and apply them on data placed in memory to get corrected data.

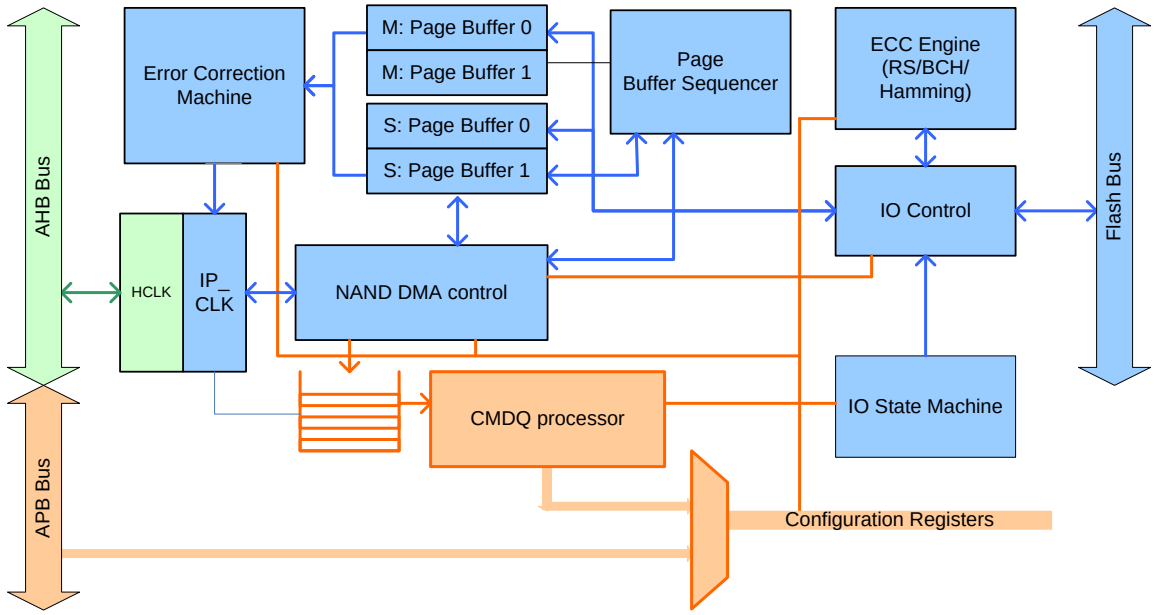
HW_ERR_CORRECTON is an optional feature version of same but implemented in hardware. I.e. hardware DMA controller will process this information and apply correction on data stored in memory without CPU involved like in pure software correction.

We chose to not have page buffer which can support in place correction – i.e., corrected data is read out from decoder without external correction as implemented. With this scheme, when data is transferred to memory it's always corrected, but as of now we chose not to have page buffers. Error detection is the bulk of the computing process that we thought is taken care and thus gives a fair trade-off for feature versus silicon for the targeted application space.

16.2 Functional Description

NAND Flash controller supports PIO mode and DMA mode of operations for NAND flash interface. Interrupt is generated upon a COMMAND sequence completion as programmed in the NAND_COMMAND register. Individual cycles of Command, Address, read data and write data or as a combined sequence for PROGRAM/ERASE/READ operation are supported. When ECC is enabled, data transfer cycles are extended to update the flash spare area to store the parity data information.

Figure 17 NAND Controller Block Diagram



16.2.1 Reed-Solomon/Hamming Error Correction for Main Data

Reed-Solomon or Hamming algorithm of error correction is selectable from register configuration for Main area data. Error Correction Capability (ECC) Encoder/Decoder works on 9 bit data (symbol size is 9 bits) irrespective of algorithm used or the input data bus configuration. Main Block ECC Encoder/Decoder engine supports the following error correction capabilities

Reed-Solomon selection, possible error correction capabilities are 4, 6, and 8 errors per each 512 bytes of information. Due to the nature of Reed-Solomon coding technique, each error referred above is actually a symbol error. I.e. for t=4 selection mean, 4 random locations of error, each consisting of 9 consecutive bit flips. Please refer to Table 6 below for detailed information on error correction capabilities and parity data lengths for different page sizes.

Hamming selection, possible error correction capability is 1bit error per each 512 bytes of information. Please refer to the following table for parity size information of different page sizes.

Table 52 Reed-Solomon ECC for Main area data

ECC (per 512 bytes)	Page Size (bytes)	Parity Size (bytes)
t=1	256	4 bytes {18 bits, rounded from 3 bytes}
t=4	512	12 bytes {rounded from 9 bytes}
t=6	512	16 bytes {108bits, 14bytes rounded to 16bytes}
t=8	512	20 bytes {rounded from 18 bytes}
t=4	1k	20 bytes {rounded from 18 bytes}
t=6	1k	28 bytes {rounded from 27 bytes}
t=8	1k	36 bytes
t=4	2k	36 bytes
t=6	2k	56 bytes
t=8	2k	72 bytes
t=4	4k	72 bytes

ECC (per 512 bytes)	Page Size (bytes)	Parity Size (bytes)
t=6	4k	108 bytes
t=8	4k	144 bytes

Table 53 Hamming ECC for Main area data

ECC (per 512 bytes)	NAND Flash Page Size (bytes)	Parity Size (bytes)
t=1	256	4 bytes {22 bits}
t=1	512	4 bytes {24 bits}
t=1	1024	4 bytes {24 bits X 2}
t=1	2048	16 bytes {24 bits X 4}
t=1	4096	32 byte { 24 bits X 8}

16.2.2 Hamming Error Detection/Correction for Spare Block Data

Hamming algorithm based error protection is selectable for Spare area data, which works on 12-bit data irrespective of algorithm used or the input data bus configuration. Only 1-bit errors can be corrected in Spare area with this implementation.

The following table summarizes the parity size lengths for different Tag sizes. TAG_BYTE_SIZE field in NAND_CONFIG register is always inclusive of these parity bytes when ECC for Tag data is enabled. Hamming selection for spare area is only possible with RS/Hamming selection of Main area. It's NOT supported along with BCH ECC selection described in next section.

Table 54 Hamming ECC for Tag Information in Spare Area

ECC (per total Tag bytes)	Total Tag Size (bytes)	Parity Size (bytes)
t=1	4 to 512 bytes	4 bytes

16.2.3 BCH Error Detection/Correction for Main and Spare Block Data

The following table summarizes the parity size lengths requirements for different Tag sizes

Table 55 BCH ECC for Main and Spare

ECC (per 512 bytes)	Page Size (bytes)	Parity Size (bytes)
t=4	2k	28 bytes
t=8	2k	52 bytes
t=14	2k	92 bytes
t=16	2k	104 bytes
t=4	4k	56 bytes
t=8	4k	104 bytes
t=14	4k	184 bytes
t=16	4k	208 bytes

Figure 18 Flash Organization with RS/Hamming ECC Selection

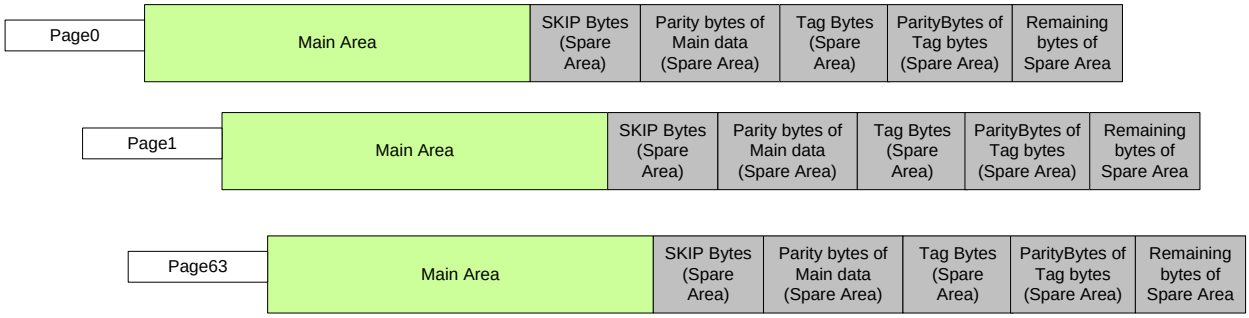
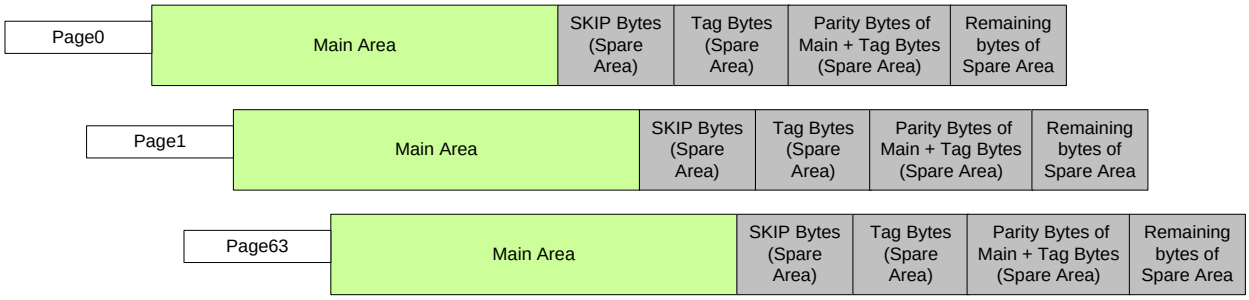


Figure 19 Flash Organization with BCH ECC Selection



16.2.4 Hardware Error Correction

The NAND controller consists of dual buffers of 4k page size to support hardware error correction. With any of the ECC algorithm selection data read from the NAND controller, the DMA interface is already corrected (if it is correctable). Data read from the Flash interface is stored in the buffer and supplied to ECC logic to locate error location and apply correction.

- **RS/Hamming ECC:** Once the page buffer is ready and error information is available correction is applied on the buffer and controller starts supplying data on DMA interface.
- **BCH ECC:** Once the page buffer is ready and error information is available controller starts supplying the data on DMA interface. As the data is read from page size buffers, correction is applied.

16.2.5 Extended Decode Status Information

Purpose is to provide detailed information of error pattern development to enable software evaluation of wear leaving and early detection of bad blocks. Once DMA completes for programmed no. of pages of read, information is available of error pattern on page basis. Without this mechanism, there is no way for software to figure out the error pattern development in media since hardware error correction is deployed. Once the errors are approaching maximum errors correctable, software can relocate these blocks to new location, so the chance of not able to recover the data errors can be ruled out.

Decode status buffer is stored with detailed information such as, sub-page indication of correctable errors, un-correctable errors. For every page with correctable/un-correctable error an entry is made to this buffer. Upon the DMA completion software can read these buffer contents through APB register interface for error pattern analysis. Please check below detailed description of related registers:

- CORRFAIL_ERR flag in NAND_ISR register indicates that, DMA transfer of specified pages resulted in correctable OR un-correctable errors. If there are no correctable/un-correctable errors in any of the pages of DMA transfer, this bit will not be SET.
- Register NAND_DEC_RESULT indicates the count of pages resulted in correctable/un-correctable errors. Software shall read these many entries from decode status buffer.

- Register NAND_DEC_STAT_BUF. Reading this register will fetch out an entry from decode status buffer. Each entry of this buffer relate to page which resulted in either correctable/un-correctable error.
- Extended decode status information is stored for all ECC selection types: RS/Hamming/BCH

Example

If there are 4 pages resulted in correctable error and 2 pages of un-correctable error out of total DMA transfer of 64 pages, PAGE_COUNT in NAND_DEC_RESULT will indicate as `PAGE_COUNT=6'. Software should read these 6 entries reading NAND_DEC_STAT_BUF register by CPU access and process it for further action.

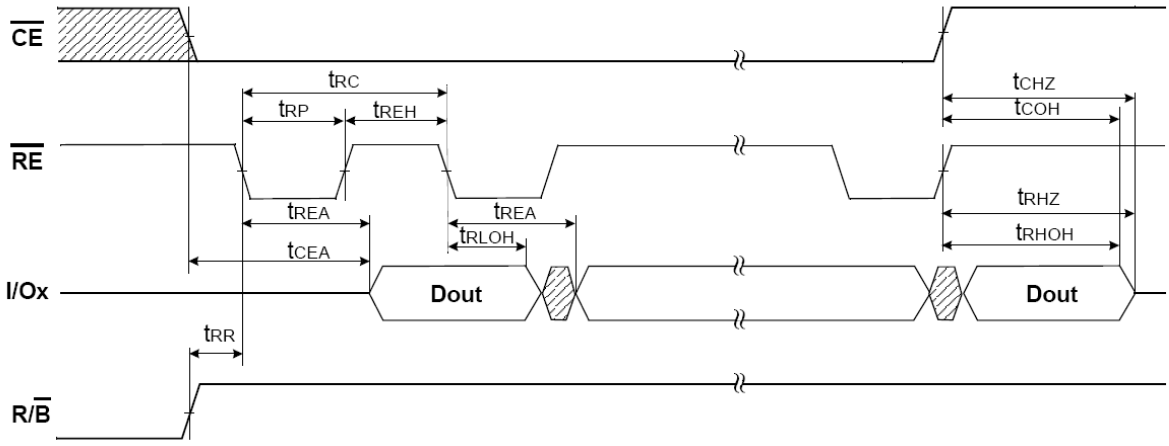
16.2.5.1 EDO mode of READ access

Latest generations of flash support the EDO (Extended Data Output) mode of operation. In this mode, Flash will drive the data for extended amount of time even after Read cycle hold time is completed. This will enable the Host system to sample the data even on completion of READ pulse.

Since the typical read cycle of operation consists of asynchronous mode of operation, (Host drives the READ pulse, and flash device drives the data), path/pad delays of read pulse will force to extend the read pulse width so that correct data is sampled on pos-edge of READ pulse. Data availability at the following neg-edge will enable the Host system application to run without extending these pulse width timings, and thus improve performance.

EDO mode of read operation controlled through the configuration bit EDO_MODE. Otherwise default Read data is sampled on pos-edge of READ pulse. Following diagram shows the extended data output timings.

Figure 20 EDO Mode Read Access

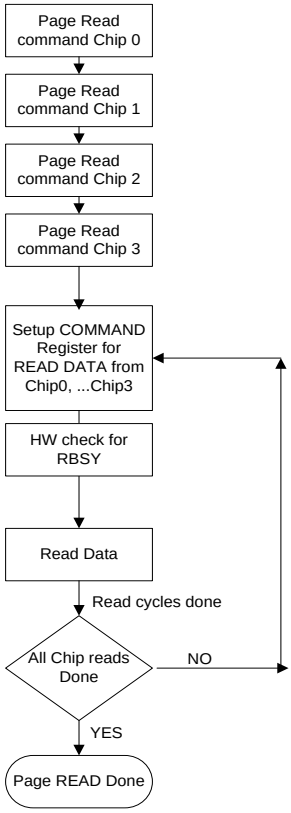


16.2.5.2 Automatic RBSY Status

Purpose is to avoid the software overhead checking the RBSY line status for CMB_RBSY application with single RBSY line. Read status command register NAND_HWSTATUS_CMD [7000:0050] is used for READ STATUS command issued by HW, configurable for the flash type used.

So when actual page read commands are issued, software can avoid the polling check for RBSY lines. When CMB_RBSY is enabled for read data transfer, hardware issues Read Status command using command byte in NAND_HWSTATUS_CMD register, and iterates till the flash is ready and moves to data read phase and completes the data transfer.

Figure 21 Interleaved READ Access for Combined RBSY Application with Auto RBSY



16.2.5.3 Skip Spare Bytes

Flash vendors can provide bad block information at the spare locations. In order to preserve these bytes for file system maintenance, the skip spare feature makes the controller skip first bytes of the spare area as selected with SKIP_SPARE_SEL. Refer to NAND_CONFIG register for different options provided. Bit [23] of NAND_CONFIG, SKIP_SPARE_EN – turns on this mode of operation with the ability to skip 4/8/12/16 bytes.

16.2.5.4 Command Queue Processor

Command queue scheme is targeted to reduce the software overhead in managing the I/O transfers on a page basis. Each IO transfer typically consists of a page size of transfer for read/write/copy back or block erase. Considering 64k block of data transfer, current approach requires setup, initiation and track of 32 I/O CMD transfers for 2k page size selection. DMA is initialized once followed by multiple commands of I/O, sometimes repeatedly just issuing the new command changing the address registers.

In command queue implementation, parameters required for each of the command (I/O transaction) are put together in the form of a packet. Software keeps on posting these packets, reducing the overhead to handle the page transfer interrupts etc. NAND I/O and state control are now fed with the configuration of the current command in progress. After the completion of current command new packet is fetched from the command queue to proceed for the new IO transfer without software intervention.

The following parameters are identified, which could vary from packet to packet depending on the flash sequences to be carried out. Each command packet consists few or all of the following parameters and each parameter is nothing but 32-bit configuration register in the existing hardware implementation. The command queue scheme with the above PACKET parameters is detailed in next section.

Table 56 Command Queue Registers

Command Queue Data set
NAND_COMMAND
NAND_CMD_REG1
NAND_CMD_REG2
NAND_ADDR_REG1
NAND_ADDR_REG2
NAND_DMA_MST_CTRL
NAND_DMA_CFG.A
NAND_DMA_CFG.B
NAND_DATA_BLOCK_PTR
NAND_TAG_PTR
NAND_ECC_PTR
NAND_HWSTATUS_CMD
NAND_HWSTATUS_MASK
NAND_CONFIG

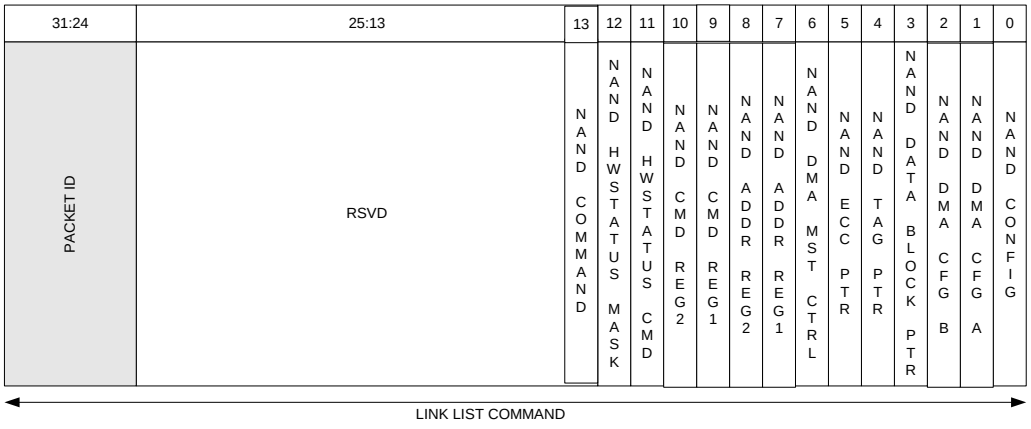
Registers not listed above are not part of command queue configuration, and static during the duration of Command queue execution. These should be properly configured as described in programming guide lines and register spec.

Command Queue Definition

Terminology:

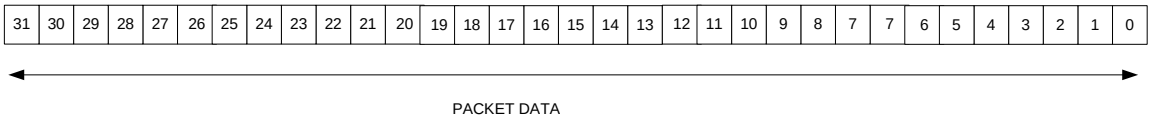
- Command Queue Command: command queue command register to define the register set programming.
- Command Queue Packet: one set of command and respective command data/registers.
- Command Queue Data: command packet parameter/register set value to be configured.
- Command Queue: set of command packets.

Figure 22 Command Queue Command Format



Within the ***command*** register, bit [13:0] is used to identify which register(s) need to be programmed with ***command queue data(s)*** that followed. A '1' within bit [13:0] indicate the corresponding register will require programming with ***command queue data***; a '0' indicate there's no programming needed for the corresponding register. The command queue command parser will always scan the ***command queue command*** starting from bit [0] to bit [13].

Figure 23 Command Queue Data Format



Following the ***command queue command***, one or more ***command queue data(s)*** will follow. The 1st ***command queue data*** following the ***command queue command*** corresponds to the right most '1' bit within bit [13:0] of the ***command queue command***. Similarly, the 2nd command queue data following the command queue command corresponds to the 2nd right most '1' bit within bit [13:0] of the ***command queue command***.

Figure 24 Command Queue Packet Format

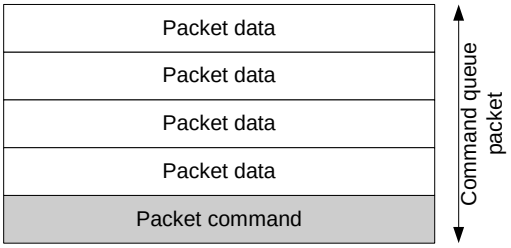
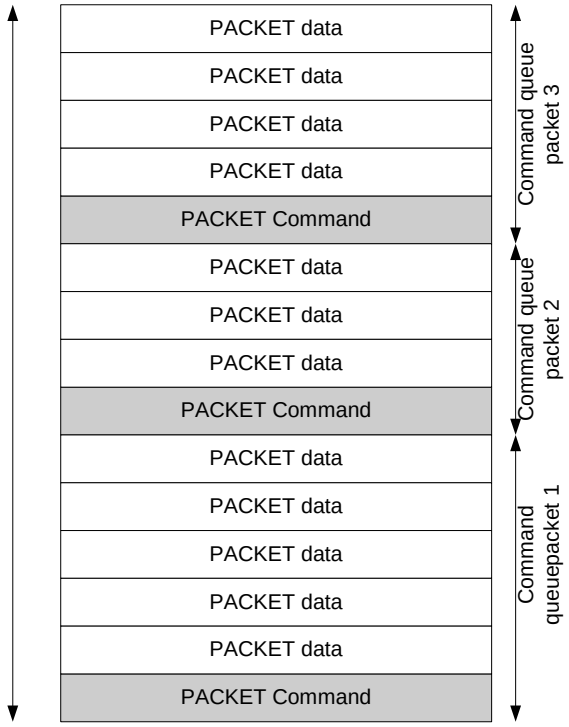


Figure 25 Command Queue Format

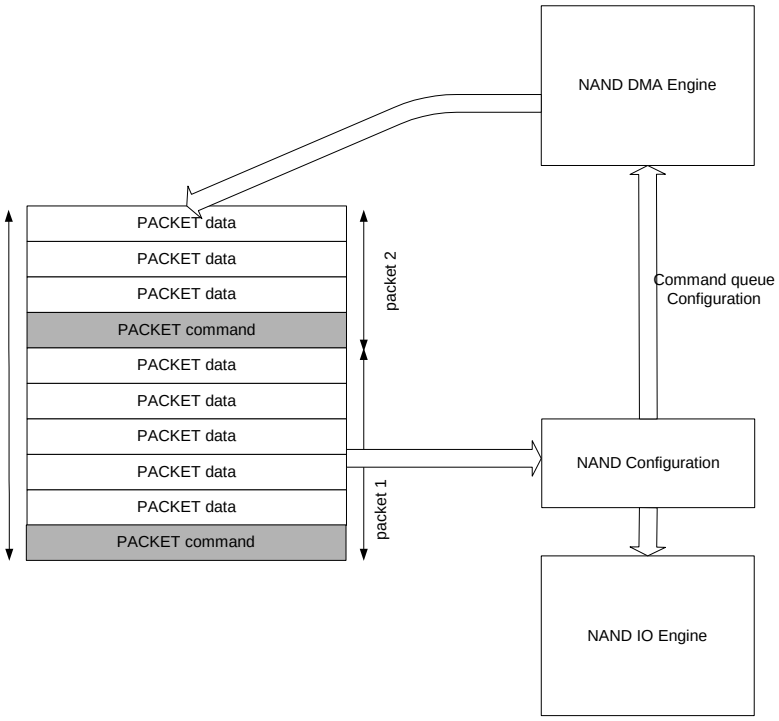


Command Queue Parser

Command queue parser will fetch one command followed by respective packet data fetches from this internal buffer as many as bit fields set in PACKET command register. When all the PACKET command data are fetched, corresponding sequence of

register settings will configure the NAND controller engine for specific sequence of IO operation. As soon as one command packet is completed, new command packet is processed.

Figure 26 Command Queue Parser



16.2.5.5 Memory Lock

The hardware provides a programmable address range on the flash controller, and a write once unchangeable bit for each range (unless the HW is reset) to implement this feature for software.

NAND_LOCK_CONTROL: software configures which range check is applicable now.

NAND_LOCK_STATUS: HW indicates which of the range is matched caused this interrupt.

NAND_LOCK_APERTURE_START0 – NAND_LOCK_APERTURE_END0: If Block erase command address matches this range (inclusive of start, ends) , actual flash erase is skipped and LOCK_ERROR interrupt is generated.

16.3 Programming Guidelines

NAND flash controller register interface allows for a flexible programming sequence that is fully under firmware control. Firmware can issue command, address and data transfers individually or combine them to form Read/Write/Status read sequences that the NAND flashes require.

For all the programming sequences described below, it is assumed that:

- NAND_TIMING register has already been programmed with the correct value for the NAND device in use. More details on programming the NAND timing register are described later in this section.
- BUS_WIDTH field is set or cleared based on the flash card data width requirement.

Interrupt Behavior for WRITE Transfers:

DMA done interrupt, IS.DMA_DONE in NAND_DMA_MST_CTRL will be set and which happens before the Command completion for a single page DMA transfer size.

Command done interrupt, IS.CMD_DONE as in NAND_ISR indicates the I/O transaction completion as per NAND_COMMAND register configuration.

If spare access with B_VALID or HW_ECC is enabled, CMD_DONE interrupt guarantees that all of these operations are completed.

Interrupt Behavior for READ Transfers

Command done interrupt, IS.CMD_DONE as in NAND_ISR indicates the I/O transaction completion as per NAND_COMMAND register configuration.

DMA done interrupt, IS.DMA_DONE in NAND_DMA_MST_CTRL will be set which happens after the Command completion for a single page DMA transfer size. In addition it is guaranteed that actual data are transferred to XMB destinations as programmed which is inclusive of TAG information if it is enabled.

If HW ECC is enabled, IS.ECC_ERR guarantees that the main block or TAG blocks information which corresponds to current page are transferred to XMB memory and ready for Error correction.

16.3.1 Command Transfer

- Write to the NAND_CMD_REG1 with the command byte.
- Write to NAND_COMMAND register with CLE bit set, CE bit field set to the desired Chip Enable and the GO bit set. The CLE bit being set indicates that a command needs to be issued. The GO bit being set initiates the CLE transfer on the interface.

Firmware can now check the status of the operation by polling the GO status. GO bit is cleared by the hardware on completion of the operation. Alternately, the firmware can enable the operation completion interrupt and handle the interrupt

16.3.2 Address Transfer

- Write to the NAND_ADDR_REG1/NAND_ADDR_REG2 with address bytes to be transferred. Firmware needs to form the address bytes in the format and sequence expected by the NAND device being used. Order of the address transfer will be NAND_ADDR_REG1 [7:0], NAND_ADDR_REG1[15:8], NAND_ADDR_REG1[23:16], NAND_ADDR_REG1[31:24], NAND_ADDR_REG2[7:0], NAND_ADDR_REG2[15:8], NAND_ADDR_REG2[23:16], NAND_ADDR_REG2[31:24]. Actual number of address bytes transferred will depend on the ALE_SIZE field. A maximum of 8 address bytes can be issued.
- Write to NAND_COMMAND register with ALE bit set, CE bit field set to the desired Chip Enable and the GO bit set. The ALE bit being set indicates that address cycles are present in the current sequence. The GO bit being set initiates the ALE transfer on the interface.
- Wait for completion of the operation through GO bit polling or CMD_DONE interrupt.

16.3.3 Write Data Transfer

This is used for writing data to the NAND Flash.

- Configure the PAGE_SIZE_SEL.A, BUS_WDITH fields in the NAND_CONFIG register as required by the Page size requirement of the Flash card.
- Configure the DMA_BLOCK_SIZE.A field in the NAND_DMA_CFG.A register with the number of bytes to be transferred. This is the total number of bytes to be transferred and can be up to 64KB. If the DMA block size is less

than the page size then those bytes have been transferred, the operation will be indicated as complete and the appropriate status/interrupt bits will reflect completion.

- Configure the NAND_DATA_BLOCK_PTR register with the data block address to be written to the flash.
- The above configuration initiates data transfers to flash Device. However, transfers to the NAND device will start as soon as the data are available in FIFO. Data transfers are throttled based on the FIFO full condition.
- Program the NAND_DMA_MST_CTRL register DIR, BURST SIZE fields for transmit and enable the DMA_GO bit.
- Write to NAND_COMMAND register with the TX bit set, the CE bit field set of the desired chip and the GO bit set.

This operation will start the write transfers on the NAND interface. Data is removed from the FIFO in little endian order and transferred on the interface. Individual transfer is a byte or half-word wide based the BUS_WIDTH parameter programmed. Transfers continue until a page worth of data is written.

16.3.4 Read Data Transfers

This is used for reading data from the NAND Flash.

- Configure the DMA_BLOCK_SIZE.A and PAGE_SIZE_SEL.A fields in the NAND_DMA_CFG.A register with the number of bytes to be transferred. This is the total number of bytes to be transferred and can be up to 64KB.
- Configure the NAND_DATA_BLOCK_PTR register with the data block address to be written to the flash.
- The above configuration's initiates data reads from the NAND to the FIFO. Data reads on the NAND interface will be throttled based on the FIFO full condition. When there is room in the FIFO then reads on the interface will be initiated and the read data will be written into the FIFO in little endian order.
- Program the NAND_DMA_MST_CTRL register DIR, BURST SIZE fields for receive and enable the DMA_GO bit. The above configuration prepares the transfer from FIFO to AHB bus. But the actual data transfer starts only after valid entries available in FIFO.
- Write to NAND_COMMAND register with the RX bit set, the CE bit field set of the desired chip and the GO bit set.

This operation will continue until the transfer size worth of data is read and placed into the FIFO as per transaction size selected. Data is removed from the FIFO by the DMA engine. Polling or interrupt can be used by firmware to determine the completion of each operation.

16.3.5 NAND Flash Operations

All sequences described below are optimized for the performance of common NAND sequences by enabling multiple control bits when issuing NAND_COMMAND. Software does have the flexibility of issuing these transfer operations individually and achieving the same results, albeit with lower performance. In the following description, "operation" is used to indicate the command issued to the NAND Flash Controller; "sequence" refers to the sequence of events on the NAND Flash Interface.

16.3.5.1 Reset Sequence

In order to reset the NAND Flash, a reset command needs to be issued.

- Write to NAND_CMD_REG1 with LSB byte as "0xFF"
- Write to NAND_COMMAND register with CLE bit set, and GO bit set. Clear all the other control bits.
- Reset command is issued on the NAND interface
- NAND Controller operation completion is reflected in the status bit (GO bit will be cleared). Additionally, command done interrupt is generated if it has been enabled.

16.3.5.2 Status Read Sequence

This sequence is used to read the status from the NAND Flash. By definition, this requires issuing a command "0x70" followed by reading a byte of data. This can be achieved in a single operation as follows:

- Write to NAND_CMD_REG1 with LSB byte as “0x70”
- Write to NAND_COMMAND register with the following
  - CLE bit set
  - PIO bit set, RX bit set
  - TRANS_SIZE set to 1 byte
  - Clear bits TX, SEC_CMD, AFT_DAT, A_VALID, B_VALID
  - Set the GO bit
- The GO bit being set initiates the Controller “operation” to issue a sequence on the NAND interface.
- NAND controller sends out the CLE with command “0x70” and follows this with a response read of one byte from the NAND device. The Controller handles all the timing requirements of the NAND device based on the parameters programmed in NAND_TIMING register. The read byte is placed into the NAND_RESP register (little endian).
- When the read is complete, NAND Controller operation is complete. This is reflected in the status bits, and command done interrupt is also generated when enabled. If polling is used, firmware can poll the GO bit. This is set by firmware to initiate the transfer and when the operation is done, hardware will clear this bit.

16.3.5.3 Page Program Sequence

This is used to write a page of data to the NAND Flash. First a write command “0x80” should be issued, followed by the destination address. Then, all the page data is written to the device. After writing the page data, programming is initiated by issuing the program command “0x10”. This sequence is achieved as follows:

- Write to NAND_CMD_REG1 with LSB byte as “0x80”
- Write to NAND_CMD_REG2 with LSB as “0x10”
- Write to NAND_ADDR_REG1/NAND_ADDR_REG2 with the correct address bytes to be transferred.
- Configure the PAGE_SIZE_SEL to select the page transfer size in NAND_CONFIG register.
- Configure the DMA_BLOCK_SIZE.A register with proper DMA transfer size required for a PAGE transfer. However for multi-page transfer this could be as much as the total DMA transfer size required.
- Configure the data block source address of read data in NAND_DATA_BLOCK_PTR register.
- Write to NAND_COMMAND register with the following
  - CLE bit set
  - ALE bit set, ALE_SIZE field is set to the appropriate number of bytes required
  - TRANS_SIZE set to page size.
  - TX bit is set and PIO bit is cleared (indicates that this is a data should be fetched from FIFO)
  - Set the SEC_CMD bit (for issuing the second command “0x30”)
  - Set AFT_DAT bit since second command needs to be issued after the data stage.
  - Set A_VALID bit since only the data is being written. Note that spare region can also be written to by setting A_VALID and providing the appropriate address and data size information.
  - CE bit field set to the desired chip.
- Enable DMA in the NAND_DMA_MST_CTRL register, along with the burst size, direction (transmit). Above configuration enables the DMA controller for transferring data to the NAND Flash FIFO under flow control
- Start IO transfer by setting GO bit in NAND_COMMAND register, When the GO operation is initiated the NAND Controller issues the following sequence on the interface: CLE (0x80) -> ALE -> DATA -> CLE (0x10). The DATA stage is throttled based on the availability of data in the FIFO.
- When entire sequence is completed, the operation done is indicated by the command done interrupt as well as hardware clear of `GO’ bit.

16.3.5.4 Page Read Sequence

This is used to read a page of data from the NAND Flash. First a read command “0x00” should be issued, followed by the destination address. Then, read is issued through command “0x30”. When the data becomes ready in the NAND Flash, the entire page is then read out. This sequence is achieved as follows:

- Write to NAND_CMD_REG1 with LSB byte as “0x00”
- Write to NAND_CMD_REG2 with LSB byte as “0x30”
- Write to NAND_ADDR_REG1/NAND_ADDR_REG2 with the correct address bytes to be read.
- Configure the PAGE_SIZE_SEL to select the page transfer size in NAND_CONFIG register.
- Configure the DMA_BLOCK_SIZE.A register with proper DMA read transfer size required for a PAGE transfer. However for multi-page transfer this could be as much as the total DMA transfer size required.
- Configure the data block source address of read data in NAND_DATA_BLOCK_PTR register.
- Write to NAND_COMMAND register with the following
  - CLE bit set
  - ALE bit set, ALE_SIZE field is set to the appropriate number of bytes required
  - TRANS_SIZE set to page size selection with MSB set
  - RX bit is set and PIO bit cleared (indicates that this is a read data should be placed FIFO)
  - Set the SEC_CMD bit (for issuing the second command “0x30”)
  - AFT_DAT bit needs to be cleared since second command needs to be issued before the data stage.
  - Set A_VALID bit since only data is being read. Spare region can also be read by issuing multiple A_VALID commands with the appropriate address and size OR by setting the B_VALID along with TAG_BYTE_SIZE in NAND_CONFIG register.
  - CE bitfield set to the desired chip.
- Enable DMA in the NAND_DMA_MST_CTRL register, along with the burst size, direction (receive). Above configuration enables the DMA controller for transferring data from the NAND Flash FIFO under flow control.
- Start IO transfer by setting GO bit in NAND_COMMAND register. When the GO operation is initiated the NAND Controller issues the following sequence on the interface: CLE (0x00) -> ALE -> CLE (0x30) -> DATA read. The DATA stage is throttled based on the availability of empty slots in the FIFO. Before entering the data stage the controller will ensure the NAND Flash is ready.
  - When entire sequence is completed, the operation done is indicated by the command done interrupt as well as hardware clear of `GO’ bit

16.3.5.5 Block Erase Sequence

Block erase requires block erase setup command to be issued (0x60), followed by Address for the block (row only), and finally the block erase command. This is achieved as follows:

- Write to NAND_CMD_REG1 with LSB byte as “0x60”
- Write to NAND_CMD_REG2 with LSB byte as “0xD0”
- Write to NAND_ADDR_REG1/NAND_ADDR_REG2 with the block address.
- Write to NAND_COMMAND register with the following
  - CLE bit set
  - ALE bit set, ALE_SIZE field is set to the appropriate number of bytes required.
  - PIO, TX, RX bits are cleared
  - Set the SEC_CMD bit (for issuing the second command “0xD0”)
  - AFT_DAT bit is cleared (0), since the second command needs to be after address stage

- Clear A_VALID, B_VALID bits
- CE bit field set to the desired chip.
- GO bit also needs to be set to initiate the IO transfer

- When the above GO operation is initiated the NAND Controller issues the following sequence on the interface: CLE (0x60) -> ALE -> CLE (0xD0).
- When entire sequence is completed, the operation done is indicated by the command done interrupt as well as hardware clear of `GO' bit

16.3.5.6 Read ID Sequence

Read ID requires issuing a READ ID command (0x90), followed by an address cycle and then reading 4 bytes of ID information. This is achieved as follows:

- Write to NAND_CMD_REG1 with LSB byte as "0x90"
- Write to NAND_ADDR_REG1 with the address byte.
- Write to NAND_COMMAND register with the following
  - CLE bit set
  - ALE bit set, ALE_SIZE field is set to the appropriate number of bytes required.
  - PIO, RX bits are set
  - TX bits is cleared
  - TRANS_SIZE field is set to 4 bytes (maximum of 8 bytes possible)
  - SEC_CMD, AFT_DAT bits are cleared
  - Clear A_VALID, B_VALID bit
  - CE bitfield set to the desired chip.
  - GO bit also needs to be set.
- When the above GO operation is initiated the NAND Controller issues the following sequence on the interface: CLE (0x90) -> ALE -> DATA READ.
- The data that is read is written into the NAND_RESP register (since PIO bit is set)
- When entire sequence is completed, the operation done is indicated by the command done interrupt as well as hardware clear of `GO' bit.

16.3.5.7 Multiple Page Write with HW ECC and Tag Information

This describes the case where multiple pages of data are to be transferred to the NAND device, along with the ECC parity and Tag information into the spare region. The controller allows for separating the data region from the spare region. These are defined as Blocks A, and B. Block A refers to the data payload (usually a page size), and Block B refers to the TAG information of spare. If the HW ECC is used, then firmware only needs to supply the tag information size for Block B and ECC configuration. Following sequence shows the required programming for transferring 4 pages of data, along with the associated tag.

- Write to NAND_CMD_REG1 with LSB byte as "0x80"
- Write to NAND_CMD_REG2 with LSB byte as "0x10"
- Write to NAND_ADDR_REG1/NAND_ADDR_REG2 with the correct address bytes to be transferred.
- Program NAND_DMA_CFG.A register with proper DMA_BLOCK_SIZE.A (4*page_size) and PAGE_SIZE_SEL fields. Configure the Data block source address in NAND_DATA_BLOCK_PTR register.
- Program NAND_DMA_CFG.B register with the total number of TAG bytes to be transferred to the spare region for all the 4 pages along with the TAG_BYTE_SIZE for a page. If the tag information is 8 bytes, then program this to (4 *

8bytes/page) = 32bytes. When ECC is enabled, Tag size for DMA configuration should not consider the parity data size of 4 bytes, since these are calculated and appended by hardware automatically.

- Configure the Tag data source address in NAND_TAG_PTR register.
- Enable HW ECC by setting HW_ECC, ECC_EN_TAG bits in NAND_CONFIG register.
- Hardware ECC Encoder will generate the parity bytes based on the ECC algorithm selection in NAND_CONFIG register and PAGE SIZE selection for main block data as well as Tag information.
- TAG_BYTE_SIZE includes the ECC parity bytes (fixed 4 bytes of parity for Tag information for Tag size up to 60 bytes) and should be word aligned. Tag information always includes the 4 bytes of ECC for Tag when ECC_EN_TAG is selected.
- Write to NAND_COMMAND register with the following
  - CLE bit set
  - ALE bit set, ALE_SIZE field is set to the appropriate number of bytes required
  - TRANS_SIZE set to page size selection with MSB bit set.
  - TX bit is set (indicates that this is a data write)
  - Set the SEC_CMD bit (for issuing the second command “0x30”)
  - Set AFT_DAT bit since second command needs to be issued after the data stage.
  - Set A_VALID, B_VALID. A_VALID being set indicates that this operation requires data payload transfer, B_VALID being set indicates that spare access required for Tag information.
  - CE bit field set to the desired chip.
- Enable the DMA controller for transferring data to the NAND Flash FIFO under flow control with proper direction DMA_DIR and BURST_SIZE settings. This will trigger transfers to both the data FIFO and the tag FIFO. Spare region access starts with main block ECC data followed by Tag.
- Start IO transfer by setting GO bit in NAND_COMMAND register. When the GO operation is initiated the NAND Controller issues the following sequence on the interface: CLE (0x80) -> ALE -> WR DATA -> WR ECC Parity Bytes -> WR Tag Info -> WR Tag ECC bytes -> CLE (0x10). The DATA/ECC/Tag stage is throttled based on the availability of data in the respective FIFO’s or ECC computations.
- When entire sequence is completed, the operation done is indicated by the command done interrupt as well as hardware clear of `GO’ bit.
- Since the DMA transfer size was programmed for 4 pages, the DMA is constantly filling the FIFO with the next page data (and similarly the tag information). Therefore, the next page data is readily available in the FIFO at the end of the prior sequence.
- Firmware issues the next page operation by programming the NAND_COMMAND register as shown above. This process is repeated for the remaining two pages. For this entire transfer of 4 pages, one DMA setup is required, followed by 4 commands being issued by writing to the NAND_COMMAND register for every page.

16.3.5.8 Multiple Page Read with HW ECC and Tag Information

This describes the case where multiple pages of data are to be read from the NAND device. ECC will be computed on the read data and location information to correct error bytes will be identified. Along with this the remaining tag information in the spare region (only) will be read to memory. The data and the spare region data are separated into two FIFO’s. These are defined as Blocks A and B. Block A refers to the data payload (usually a page size), and Block B refers to the Tag information of Spare region contents. Following sequence shows the required programming for transferring 4 pages of data, along with the associated tag.

- Write to NAND_CMD_REG1 with LSB byte as “0x00”
- Write to NAND_CMD_REG2 with LSB byte as “0x30”
- Write to NAND_ADDR_REG1/NAND_ADDR_REG2 with the correct address bytes to be read.

- Program NAND_DMA_CFG.A register with proper DMA_BLOCK_SIZE.A (4*page_size) and PAGE_SIZE_SEL fields. Configure the Data block destination address in NAND_DATA_BLOCK_PTR register.
- Program NAND_DMA_CFG.B register with the total number of TAG bytes to be transferred to the spare region for all the 4 pages along with the TAG_BYTE_SIZE for a page. If the tag information is 8 bytes, then program this to (4 * 8bytes/page) = 32bytes. TAG_BYTE_SIZE calculation for DMA configuration should consider the 4 bytes of parity. i.e for 60 bytes of tag size, 64 byte size should be considered.
- Configure the Tag data source address in NAND_TAG_PTR register.
- Enable HW ECC by setting HW_ECC, ECC_EN_TAG bits in NAND_CONFIG register.
- TAG_BYTE_SIZE programmed value includes the ECC parity bytes (fixed 4 bytes of parity for Tag information for Tag size up to 60 bytes) and should be word aligned. Tag information always includes the 4 bytes of ECC for Tag when ECC_TAG_EN is selected.
- Write to NAND_COMMAND register with the following
  - CLE bit set
  - ALE bit set, ALE_SIZE field is set to the appropriate number of bytes required
  - TRANS_SIZE set to page size.
  - RX bit is set (indicates that this is a read)
  - Set the SEC_CMD bit (for issuing the second command “0x30”)
  - AFT_DAT bit needs to be cleared since second command needs to be issued before the data stage.
  - Set A_VALID and B_VALID. A_VALID being set indicates that this operation requires data payload transfer and B_VALID indicates the Tag information to be transferred.
  - CE bit field set to the desired chip.
- Enable the DMA controller for transferring data to the NAND Flash FIFO under flow control with proper direction (receive) and burst size settings. This will prepare transfers from both the data FIFO and the tag FIFO to the system. ECC FIFO is updated by the HW Decoder engine after page size READ transfer is completed.
- Start IO transfer by setting GO bit in NAND_COMMAND register. When the GO operation is initiated the NAND Controller issues the following sequence on the interface: CLE (0x00) -> ALE -> CLE (0x30) ->RD DATA -> RD ECC Parity Bytes -> RD Tag Info. The DATA/Tag stage is throttled based on the availability of space in the respective FIFO’s.
- When entire sequence is completed, the operation done is indicated by the command done interrupt as well as hardware clear of `GO’ bit.
- If there is an ECC error, this is indicated to firmware through an IS.ECC_ERROR interrupt. Firmware needs to read the error information and apply correction to the data in the main memory. Firmware should snap the ERR_PAGE_NUMBER and ERR_COUNT.A information available at this time from NAND_ISR register for a multi-page transfer.
- Firmware issues the next page read operation by programming the NAND_COMMAND register as shown above. This process is repeated for the remaining two pages. For this entire transfer of 4 pages, one DMA setup is required, followed by 4 commands being issued by writing to the NAND_COMMAND register after every page.
  - Note that Firmware need not wait for IS.ECC_ERROR interrupt to proceed for a new read transfer with or without using the ECC pipeline feature.

16.3.5.9 Interleaved transfers with combined RBSY

R/BSY from all the flashes can be wired to from a single RBSY to reduce the pin usage in some applications. Please follow the programming sequences explained below for read/write transfers to achieve performance close to the case of R/BSY signals available from each of the flash connected

WRITE Transfers

- Issue Write commands to each card by checking the R/BSY status from READ RESPONSE commands
- Write commands here refer to CMD + DATA transfers as single command sequence of NAND controller. NAND controller will not look for RBSY of line to proceed for initiating commands followed by data transfers in combined RBSY case.
- In individual R/BSY case, these commands can be issued without checking the R/BSY of each card and NAND controller takes care of starting the transfers only after card is READY.

The only difference now, is software will check the READY state by issuing the READ RESPONSE commands before initiating any new transfers.

READ Transfers

- Similar to interleaved transfers, configure for CLE & ALE cycles only while issuing commands to each card connected. Do not include data transfers for any of these commands. Because NAND controller hardware is supposed wait for card ready state before generating the READ DATA cycles.
- Now commands being issued, R/BSY of each of these cards is de-asserted. Since it's combined R/BSY case, not READY on any line is seen as READY BSY low by the controller.
- Continue with the READ DATA transfers, for any of the card connected further to rest of the cards.
- DATA READ transfers will start as soon as ALL OF CONNECTED FLASHES is ready. In individual R/BSY case, it will start as soon as the respective is READY. In the interleaved operation there will be very little impact on performance because of the following reason.
- Considering RBSY latency time of 25us, the performance hit will be for first card only. By the time READ DATA transfers on this are completed, all the rest of the RBSY latencies time should have been elapsed, and waiting for DATA transfer cycles.

16.3.5.10 PIO Write data transfers

This describes the case of write transfers for page program in PIO mode of operation. Following sequence on the interface is required in the interface to achieve this program sequence:

CLE (0x80) -> ALE -> DATA -> CLE (0x10)

This sequence can be viewed as multiple GO sequences in PIO mode such as:

- Single GO command sequence for combined CLE, ALE transfers for initialization
- Multiple GO commands to transfer data in PIO mode
- Final program command

Following programming sequence illustration assumes a page size of 512bytes without spare area access. For spare area access data sequence as in Step 2 should be repeated for required transfer length.

- Write to NAND_CMD_REG1 with LSB byte as “0x80”
- Write to NAND_ADDR_REG1/NAND_ADDR_REG2 with the correct address bytes to be transferred.
- Write to NAND_COMMAND register with the following
  - CLE bit set
  - ALE bit set, ALE_SIZE field is set to the appropriate number of bytes required
  - Set the SEC_CMD bit (for issuing the second command “0x10”)
  - AFT_DAT bit needs to be cleared since second command needs to be issued before data cycles.
  - CE bit field set to the desired chip.
  - GO bit is set, which initiates the IO transfer and gets cleared on transfer completion.

- Write to NAND_RESP register with appropriate data bytes to be transferred. LSB byte in this register is the first one to be transmitted out. NAND_RESP registers is a shared register used for PIO write data/PIO read data. Please note that NAND_RESP write data is not readable by software
- Write to NAND_COMMAND register with the following
  - Clear any other bit fields which are SET earlier
  - TRANS_SIZE set to select the small transaction sizes and configure the TRANS_SIZE to be 0x3 for 4 byte transfer. In PIO mode only 4 bytes can be transferred with the single GO command
  - TX bit is set and PIO bit is set which indicates that this is a data should be fetched from NAND_RESP register.
  - Set A_VALID bit since only the data is being written. Note that spare region can also be written to by setting A_VALID and provide the appropriate address and data size information.
  - CE bit field set to the desired chip.
  - GO bit is set, which initiates the IO transfer and gets cleared on transfer completion
- Repeat the data transfer sequence multiple times(128 times for total 512 byte transfer) setting the GO bit to complete the total page transfer.
- Write to NAND_CMD_REG1 with LSB as “0x10”
- Write to NAND_COMMAND register with the following
  - Clear any other bit fields which are SET earlier
  - CLE bit set
  - CE bit field set to the desired chip.
  - GO bit is set, which initiates the IO transfer and gets cleared on transfer completion.

16.3.5.11 PIO Read Data Transfers

This describes the case of read transfers for page read in PIO mode of operation. Following sequence on the interface is required in the interface to achieve this sequence:

CLE (0x00) -> ALE -> CLE (0x30) -> DATA

This sequence can be viewed as multiple GO sequences in PIO mode such as,

1. Single GO command sequence for combined CLE (0x00), ALE and CLE (0x30) transfers for initialization
2. Multiple GO commands to transfer data in PIO mode.

Following programming sequence illustration assumes a page size of 512bytes without spare area access. For spare area access data sequence as in Step 2 should be repeated for required transfer length.

- Write to NAND_CMD_REG1 with LSB byte as “0x00” and NAND_CMD_REG2 with LSB byte as “0x30”
- Write to NAND_ADDR_REG1/NAND_ADDR_REG2 with the correct address bytes to be transferred.
- Write to NAND_COMMAND register with the following
  - CLE bit set
  - ALE bit set, ALE_SIZE field is set to the appropriate number of bytes required
  - CE bit field set to the desired chip.
  - GO bit is set, which initiates the IO transfer and gets cleared on transfer completion.
- Write to NAND_COMMAND register with the following
  - Clear any other bit fields which are SET earlier
  - TRANS_SIZE set to select the small transaction sizes and configure the TRANS_SIZE to be 0x3 for 4 byte transfer. In PIO mode only 4 bytes can be transferred with the single GO command
  - RX bit is set and PIO bit is set which indicates that this is a data should be fetched from NAND_RESP register.

- Set A_VALID bit since only the data is being read. Note that spare region can also be read by setting A_VALID and provide the appropriate address and data size information.
- CE bit field set to the desired chip.
- GO bit is set, which initiates the IO transfer and gets cleared on transfer completion

- Read data is available in NAND_RESP register after the GO command completion.

Repeat the data transfer sequence multiple times(128 times for total 512 byte transfer) setting the GO bit to complete the total page transfer.

16.3.6 ONFI 1.0 Flash Support

ONFI 1.0 flash interface is slightly different:

- Chip detection - Following the RESET command, issue Read ID command with address 20, which returns ONFI signature to conclude chip detection for ONFI flash.
- Requires target initialization with Read parameter page after device discovery/chip detection procedure - which returns capabilities, features and operating parameters of device. It's not very clear from the spec, but it appears that for Power on Read Target initialization may not be required. However for program/erase operations can't be done without Target initialization
- Bad block information - available on first page or second page of a block and could happen anywhere in byte location rather than fixed location in `Legacy' flash for a specific flash vendor. Entire Page (start, end of each block) has to be read to prepare bad block table.
- Independent data buses can't be supported in hardware- which requires Host ability to drive data selectively on each of the 8-bit bus. Chip selects and control signals might be wired together for such a use case
- Read ID in `legacy' flash is equivalent to - Read ID + Read unique ID in ONFI flash.
- Power on boot - NAND timings should be programmed to Timing mode0, which is the default mode of operation after initialization
- Read Status information - is now elaborated and requires some changes
- PLUS general software changes to support all new commands starting from switching to fast transfer mode to interleaved operations.

ONFI 1.0 Flash Command Sequences

The following is required from software to support ONFI 1.0 Flash.

1. Read Status Enhanced command: CLE + ALE (3cycles) + Without RBSY check read data
2. Read Parameter Page: CLE + ALE + RBSY check and read data
3. Read Unique ID: same as Read Parameter page
4. Read Cache Enhanced: same as before command values might be different
5. Read Cache End: same as before command values might be different
6. Copy back program: CLE1 + ALE + CLE2 + RBSY check read data, Note that for dataout read cycles have to be generated.
7. Change Read Column: same as Read operation in legacy flash
8. Change Write Column: same as Random data operation in legacy flash
9. Get Features: yes, same as Read operation in legacy flash
10. Set Features - yes, CLE + ALE + write parameter data (4cycles)

16.3.7 NAND Timing Registers

Hardware implementation of timings in NAND Flash controller follows the simple equation of running the timing counter for (N+1) clocks for programmed value of N. Following equation represents the same, and a timing calculation example for typical flash timings is shown below. For different Flash timings or for different NAND Clock frequency option these have to be calculated.

- tWP (in ns) = (TWP count programmed + 1) X tNAND_CLK (Nand Clock time period)
- tREA = 20ns
- tWB = 100ns
- tRR(Max of tCR/tAR/tRR) = 20ns
- tWHR = 60ns
- tSETUP/tCS = 0ns
- tWH = 15ns
- tWP = 25ns
- tRH = 15ns
- tRP = 25ns

Considering NAND Clock source options is selected to run at system PLL frequency which is running at 100MHz, timing counts programmed will be as:

- TREA = 2
- TWB = 9
- TRR = 1
- TWHR = 5
- TSETUP= 0
- tWH = 1
- tWP = 2
- tRH = 1
- tRP = 2

16.3.8 Restrictions

- DMA_PERF_EN/HW_ERR_CORRECTION/REUSE_BUFFER Configuration Bits
  - DMA_PERF_EN=0/1, doesn't alter controller hardware behavior.
  - HW_ERR_CORRECTION = 0/1, doesn't alter controller hardware behavior. Software error correction scheme with HW_ERR_CORRECTION = 0, is deprecated in Tegra 2 Processor.
  - REUSE_BUFFER = 0/1, doesn't alter controller hardware behavior.
- Programming requirements for PAGE_SEL in NAND_COMMAND, TRANS_SIZE in NAND_CONFIG:

  PAGE_SEL configuration in NAND_CONFIG register and TRANS_SIZE in NAND_COMMAND register determines the data transfer size of any command sequences. These should be properly set before issuing GO or DMA_GO are set. Following illustration of single DMA configuration of usage explains one of this typical scenario and above said programming requirement applies.

  Page size = 512 bytes

  No of pages = 8

DMA length is configured for full length of 8 pages, such that DMA is configured only once for all 8 page transfers. For `program' operation, at the end of each page software may issue the READ RESPONE command to check the program status.

Software changes the TRANS_SIZE configuration for the READ RESPONSE command. The above restriction applies here and TRANS_SIZE should be set before issuing the new GO command to initiate the subsequent page program operations

- Programming requirements for changing the DMA_DIR:

  DMA_DIR configuration in NAND_DMA_MST_CTRL register should not be changed until the completion of IO transfer or CMD_DONE interrupt is set. Usually it happens that for write transfers, DMA transfer is completed earlier to the actual IO transfer to flash device. Since DMA_DIR controls the read/write direction for shared FIFO, DMA_DIR changes during this time will corrupt and cause data mismatch. Software should wait for CMD_DONE completion before setting up DMA for new data transfers.

- Programming requirements for non-pipelined READ transfers:

  For non-pipelined read data transfers i.e. with PIPELINE_EN = 1'b 0 configuration in NAND_CONFIG register, with ECC enabled DMA configuration should be done for every page. I.e. it requires DMA setup for each page of transfer, however for non-ECC read data transfers or all write transfers with/without ECC single DMA configuration still works.

  ECC decode works like a pipelined operation turning on PIPELINE_EN feature. There would be around 10% of performance lost if DEC_PIPELINE_EN is not used for DMA transfers in single DMA configuration. If DMA is setup on page basis {i.e. DMA length is same as page size}, DEC_PIPELINE_EN enable/disable will not have any effect.

- Command queue mode stall/disable interpretation:

  Removing LL_START is allowed at any time during command queue execution and this is a way of disabling the command queue execution. Word counter status registers in NAND_LL_STATUS register will be useful for calculating the next start address, however note that enabling the LL_START will not resume execution and will repeat the execution from start. Software should change the NAND_LL_PTR for necessary for any correction and can continue execution enabling LL_START.

- Status read commands should not be mixed up the DMA transfers in command queue mode:

  For single DMA programming of multi-page read transfers, flash status read commands can't be mixed up with command packets of data read. After full length of DMA is done, status read command packets can be formed in Command Q mode to check the status. Please note that this restriction applies to page read data [DMA write] transfers only, and write transfers it is okay to mix up with status read commands.

- Single GO operation of "Read Command2 Cache" + "Read Data" execution is not supported:

  For "Cache Read2" command, in fact any future command other than page program command2, block erase, read command2, and page program command2 cache are not supported. Software has to split the execution by doing independent commands of

  - Read Command2 Cache transfer
  - Read data transfer with RBSY as applicable.

- Truncated page transfers are not supported in APB/Command queue mode of operation:

  In APB/Command Q mode of execution, total DMA transfer length should be integer multiple of page size for main page. Please note that truncated sizes are not supported for Tag transfers. I.e. for multi pages, Tag Byte * no. of pages should equal to DMA length of Tag.

- DMA transfer size non-multiple of 4 bytes (32bit word) is not supported for data and tag information:

- Command queue stall timeout guidelines:

  NAND controller command GO [NAND_COMMNAD] and DMA_GO [NAND_DMA_MST_CTRL] & DMA have to be de-asserted if CMD_DONE interrupt is not generated within time out period as calculated below.

  Time out = Read/Write access time & no. of reads/writes per page * no. of command packets

Read/Write access time = no. of clocks per read/write access * clock period

- BCH mode Error correction
  - Error correction with involving spare only transfers is not supported. ECC calculation of last sub-page includes tag data in spare area.
  - Error correction with Main only transfers is supported.
  - Maximum possible length of Tag data size is 252 bytes.
  - Page sizes supported with BCH are: 2K, 4K only

16.3.8.1 Command Queue Sequencer Usage for Flash Transfers

In order to extend the flexibility of controller operation in command queue mode for any sequence of flash operations, RBSY_CHK, RD_STATUS_CHK are added. Software can configure these fields in NAND_COMMAND register according to operation required.

As per the command queue sequence hardware automatically waits for RBSY from flash irrespective of CMB_RBSY system configuration if RBSY_CHK is enabled in NAND_COMMAND register. And before proceeding to any new command sequence hardware reads the status from flash if RD_STAUS_CHK is enabled and will flag out LL_ERR and command queue execution will be aborted. As an example, for Cache program commands when READ STATUS information is valid only when page buffer ready is asserted. Please refer to TOSHIBA MLC auto program with cache sequence for the same. Also in multi-page cache/without cache program before the second plane data is loaded READ STATUS information is invalid.

Overall, there are cases where READ STATUS command check is not required as well. Programmable options for RBSY_CHK/RD_STATUS_CHK will thus generalize the usage to support any type of transactions even for future scenarios. For all of the operations listed below NAND controller wait for RBSY or checking the previous operation status is programmable using these configuration bits - RBSY_CHK/RD_STATUS_CHK for respective command.

RBSY_CHK/RD_STATUS_CHK

- READ transfers in single card/interleaved mode

  As many operations as required can be queued up. The order of writing in to XMB buffer will be in the order in which these read transactions are queued up in the buffer.

- WRITE transfer single card/interleaved mode

  As many operations as required can be queued up. Hardware waits for RBSY irrespective combined/individual RBSY mode. The order of reading XMB buffer will be in the order in which these write transactions are queued. Software should make sure to program proper READ STATUS command in NAND _HWSTATUS_CMD register following page `program' operations. It could be as well for new `program' or `cache program' commands. It is important to note that for any type of `program' commands at the end of queue read status commands have to be posted with RBSY_CHK so that NAND controller will execute these commands and thus making sure that `program' operation results are verified.

- READ STATUS commands

  In addition to the read status commands done automatically by hardware with RD_STATUS_CHK enabled, simple read status command can also be configured as a transaction.

- COPY BACK PROGRAM transfers

  Two step sequences of COPY BACK and PROGRAM should back to back for each of the card in the command queue, there is nothing differently required to be done. Programming proper value in NAND_HWSTATUS_CMD register should be done just as for any other `program' operations.

- Multi-plane page program

Two step sequences of `dummy program' and `program' should be back to back in the command queue for the same card. Programming proper value in NAND_HWSTATUS_CMD register should be done as for any other `program' operations.

- Multi-plane block erase

  For some of the cards, there are separate READ STATUS commands (F1h/F2h) to be used check the flash status. FW should configure proper Command value (K9K8G09U0M series is an example which does require different command value of read status command) for the following transfers so that hardware can use the command value to obtain the status and flag in case of unsuccessful block erase operation.

- Random Data Output transfers

  Required sequences of operation:

  a. Read setup plus command with proper column address plus truncated length read
  b. Random output command with proper column address plus remaining length read

  Above two sequences should be back to back in command queue.

- Random Data Input transfers

  Required sequences of operation:

  a. Write setup with proper column address plus data cycles only and no `program' command
  b. Random data Input command with proper column address with Data cycles plus `program' command.

  Above two sequences should be back to back in the queue. Programming proper value in NAND_HWSTATUS_CMD register should be done as for any other `program' operations.

- READ ID transfers

  It is possible to queue up a transaction for read status information; however this usage may not be useful in this mode of operation.

- CACHE READ transfers

  Required sequences of operation,

  a. Read setup command with row/column address, and READ command
  b. Cache read command
  c. Data read
  d. Cache read command
  e. Data read
  f. Cache read termination command
  g. Data read

  Above sequences should be back to back in command queue.

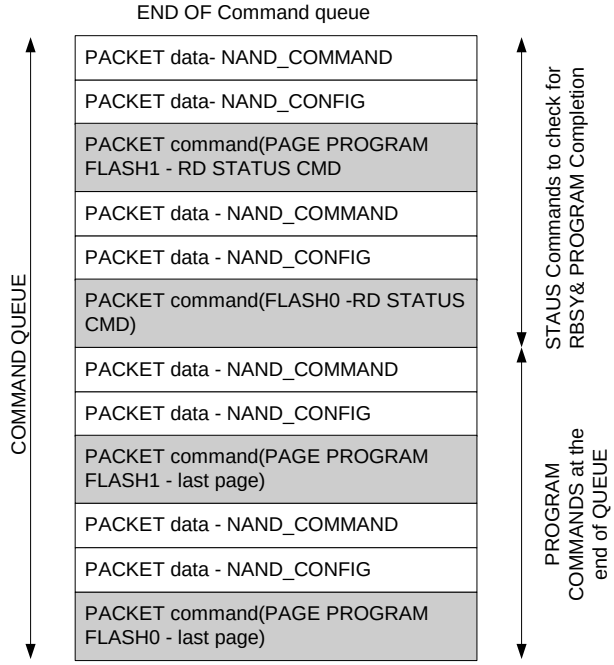
- CACHE PROGRAM transfers

  Required sequences of operation:

  a. Write setup command with row/column address, write data and write cache command
  b. Write setup command with row/column address, write data and write cache command
  c. …..
  d. Write setup command with row/column address, write data and program command

  Above sequences should be back to back in command queue. For multi-page program with cache requires 71'h for READ STATUS information. So for next flash operation to the same flash card NAND_HWSTATUS_CMD should be programmed accordingly. Also if there a chance of no more flash operations, READ STATUS command has to be queued up as well, otherwise hardware cannot figure out this status automatically. Please check the following example for reference.

Figure 27 Page Program Operations at the end of Link list



- Error conditions and PACKET ID status:

  When there is failure in any previous command sequence operation, Command queue execution will be aborted triggering interrupt to processor, if RD_STATUS_CHK is enabled. It is possible that:

  - Current Register status will not reflect the register configured which caused the failure.
  - Software has to figure out what is the last command sequence to the flash which caused this failure.
  - Re arranges the list as per application, and configures the Command queue configuration registers to re-start new sequence of operations.

16.3.8.2 Command Queue Abort/Stall Conditions

If RD_STATUS_CHK fails for any of the command packet execution is stopped right away, i.e. at this time the command packet which needs the RD_STAUS_CHK is read from the buffer and executed up to the RD_STATUS_CHK only. Associated read/write/command transaction to flash as programmed NAND_COMMAND register is abandoned further and software has to come back from this packet for recovery procedure.

Also software can halt the command queue execution at any point of time by clearing the LL_START bit of NAND_LL_CONFIG register. Removing LL_START is allowed at any time during command queue execution and is interpreted as a way of disabling the command queue execution. By this time recognizing the LL_START clear, NAND controller should have extracted next packet command and accordingly NAND_LL_STATUS registers will reflect the status as in Abort condition.

Word counter status registers in NAND_LL_STATUS register will be useful for calculating the next start address for both cases. However in case of Halt, please note that enabling the LL_START will not resume execution and will repeat the execution from start. Software should configure NAND_LL_PTR register for necessary correction and can continue execution enabling LL_START.

Next start address after Abort/Halt should be calculated as:

*NAND_LL_PTR + (LL_LENGTH_DONE – LL_LENGHT_LAST_PACKET) * 4 = Byte Address of the command packet where the command queue execution stopped either because of the COMMAND QUEUE Error [RD_STATUS_CHK error] or COMMAND QUEUE Stall.

16.4 NAND Registers

16.4.1 NAND_COMMAND_0

Configuration of this register determines the flash sequence to be initiated by NAND controller.

NAND Command Register

Offset: 000h | Read/Write: R/W | Reset: 0b00000000100000000000000000000100

Bit	Reset	Description
31	0x0	GO: Flash sequence enable. Hardware clears when programmed IO operation is completed. 0 = DISABLE 1 = ENABLE
30	0x0	CLE: Flash sequence Command cycle enable 0 = DISABLE 1 = ENABLE
29	0x0	ALE: Flash sequence Address cycle enable 0 = DISABLE 1 = ENABLE
28	0x0	PIO: PIO mode of operation enable 0 = DISABLE [Data transfer is to/from system memory as pointed by DMA destination registers] 1 = ENABLE [Data transfer is to/from NAND_RESP/RESP2 registers
27	0x0	TX: Write data transfer enable – Required for FLASH program operation. 0 = DISABLE 1 = ENABLE
26	0x0	RX: Read data transfer enable Required for FLASH read operation. 0 = DISABLE 1 = ENABLE
25	0x0	SEC_CMD: CMD2 sequence enable 0 = DISABLE 1 = ENABLE
24	0x0	AFT_DAT: CMD2 placement control 1 - 0 - 0 = DISABLE[CMD2 CLE cycle is issued right after Address transfer cycles, typical usage during FLASH read] 1 = ENABLE [CMD2 CLE cycle is issued after data transfer cycles. This is the typical usage during FLASH program operation]
23:20	0x8	TRANS_SIZE: Transfer size of bytes depends on PAGE_SIZE_SEL field of NAND_CONFIG register 0 = BYTES1 1 = BYTES2 2 = BYTES3 3 = BYTES4 4 = BYTES5 5 = BYTES6 6 = BYTES7 7 = BYTES8 8 = BYTES_PAGE_SIZE_SEL

Bit	Reset	Description
19	0x0	A_VALID: Main data region transfer enable0 = DISABLE 1 = ENABLE
18	0x0	B_VALID: Spare region (aka TAG) transfer enable 0 = DISABLE 1 = ENABLE
17	0x0	RD_STATUS_CHK: Hardware assisted read status check enable 1. 0 = DISABLE 1 = ENABLE, Indicates to controller that current IO sequence need Read Status check condition to be qualified. Please refer to NAND_HWSTATUS_CMD register for qualifier condition
16	0x0	RBSY_CHK: Hardware assisted rbsy check enable 1. 0 = auto RBSY check is disabled notes: please refer to NAND_HWSTATUS_CMD register for qualifier condition 0 = DISABLE 1 = ENABLE, = Indicates to controller that current IO sequence need RBSY check condition to be qualified. Please refer to NAND_HWSTATUS_CMD register for qualifier condition
15	0x0	CE7: Chip select7 enable 0 = DISABLE 1 = ENABLE
14	0x0	CE6: Chip select6 enable 0 = DISABLE 1 = ENABLE
13	0x0	CE5: Chip selec5 enable 0 = DISABLE 1 = ENABLE
12	0x0	CE4: Chip select4 enable 0 = DISABLE 1 = ENABLE
11	0x0	CE3: Chip select3 enable 0 = DISABLE 1 = ENABLE
10	0x0	CE2: Chip select2 enable 0 = DISABLE 1 = ENABLE
9	0x0	CE1: Chip select1 enable 0 = DISABLE 1 = ENABLE
8	0x0	CE0: Chip select0 enable 0 = DISABLE 1 = ENABLE
7:6	0x0	RSVD
5:4	0x0	CLE_BYTE_SIZE: Command cycle byte count 0 = CLE_BYTES1 1 = CLE_BYTES2 2 = CLE_BYTES3 3 = CLE_BYTES4

Bit	Reset	Description
3:0	0x4	ALE_BYTE_SIZE: Address cycle byte count 0 = ALE_BYTES1 1 = ALE_BYTES2 2 = ALE_BYTES3 3 = ALE_BYTES4 4 = ALE_BYTES5 5 = ALE_BYTES6 6 = ALE_BYTES7 7 = ALE_BYTES8 8 = ALE_BYTES9 : Reserved 9 = ALE_BYTES10 : Reserved 10 = ALE_BYTES11 : Reserved 11 = ALE_BYTES12 : Reserved 12 = ALE_BYTES13 : Reserved 13 = ALE_BYTES14 : Reserved 14 = ALE_BYTES15 : Reserved 15 = ALE_BYTES16

16.4.2 NAND_STATUS_0

Collection of NAND flash RBSY line status and other internal operations status.

NAND Status Register

Offset: 004h | Read/Write: RO | Reset: 0bxxxxxxxx00xxxxxx1

Bit	Reset	Description
15	X	RBSY7: 1 = Indicates flash7 is RDY
14	X	RBSY6: 1 = Indicates flash6 is RDY
13	X	RBSY5: 1 = Indicates flash5 is RDY
12	X	RBSY4: 1 = Indicates flash4 is RDY
11	X	RBSY3: 1 = Indicates flash3 is RDY
10	X	RBSY2: 1 = Indicates flash2 is RDY
9	X	RBSY1: 1 = Indicates flash1 is RDY
8	X	RBSY0: 1 = Indicates flash0 is RDY
7	0	WR_ACT: 1 = Indicates write cycles to flash are in progress
6	0	RD_ACT: 1 = Indicates read cycles to flash are in progress
0	1	ISEMPTY: 1 = Indicates NAND controller is in IDLE state of operation, and there are no flash/DMA transactions pending.

16.4.3 NAND_ISR_0

Interrupt status hardware behavior: All interrupt status bits are SET, respective event occurs and writes 1 to clear.

NAND Interrupt Status Register

Offset: 008h | Read/Write: R/W | Reset: 0b0xxxxxxx000000000000000xx

Bit	R/W	Reset	Description
24	RO	0x0	CORRFAIL_ERR: 1 = Correctable OR Un-correctable errors occurred in the DMA transfer without regard to HW_ERR_CORRECTION feature is enabled or not. Use extended decode results in NAND_DEC_RESULT and NAND_DEC_STATUS_EXT to figure out further action for block replacement/wear leveling during file system management for software. Covers all ECC selection: RS/Hamming/BCH modes
15	RW	0x0	IS_RBSY7: 1 = Flash7 is Ready interrupt occurred. This bit is set only when not running in COMMAND QUEUE MODE
14	RW	0x0	IS_RBSY6: 1 = Flash6 is Ready interrupt occurred. This bit is set only when not running in COMMAND QUEUE MODE
13	RW	0x0	IS_RBSY5: 1 = Flash5 is Ready interrupt occurred. This bit is set only when not running in COMMAND QUEUE MODE
12	RW	0x0	IS_RBSY4: 1 = Flash4 is Ready interrupt occurred. This bit is set only when not running in COMMAND QUEUE MODE
11	RW	0x0	IS_RBSY3: 1 = Flash3 is Ready interrupt occurred. This bit is set only when not running in COMMAND QUEUE MODE
10	RW	0x0	IS_RBSY2: 1 = Flash2 is Ready interrupt occurred. This bit is set only when not running in COMMAND QUEUE MODE
9	RW	0x0	IS_RBSY1: 1 = Flash1 is Ready interrupt occurred. This bit is set only when not running in COMMAND QUEUE MODE
8	RW	0x0	IS_RBSY0: 1 = Flash0 is Ready interrupt occurred. This bit is set only when not running in COMMAND QUEUE MODE
7	RW	0x0	IS_UND: 1 = FIFO under run interrupt occurred this should not happen in general usage, if it happens there is a potential hardware issue.
6	RW	0x0	IS_OVR: 1 = FIFO is Overrun this should not happen in general usage, if it happens there is potential hardware issue.
5	RW	0x0	IS_CMD_DONE: 1 = Command operations are completed as per NAND command register settings. This is set ONLY when not running in COMMAND QUEUE MODE
4	RW	0x0	IS_ECC_ERR: 1 = ECC error generated for following reasons. ->ecc decode resulted in uncorrectable errors in one of sector(sub-page) ->ecc decode resulted in correctable errors more than trigger level as defined in TRIG_LVL in NAND_CONFIG register. Bit is set for legacy mode of ECC selection with HW_ECC & ECC_TAG_EN only i.e. for RS/Hamming selection. This bit is not set for BCH selection
3	RW	0x0	IS_LL_DONE: 1 = Command queue execution completed
2	RW	0x0	IS_LL_ERR: 1 = One of the Command queue packet execution returned error because of Read Status Check failure

16.4.4 NAND_IER_0

NAND Interrupt Enable Register

Offset: 00ch | Read/Write: R/W | Reset: 0b00000000000000000x0

Bit	Reset	Description
19:16	0x0	ERR_TRIG_VAL: Trigger for correctable error Interrupts by main ECC RS decoder, if HW_ERR_CORRECTION feature is enabled. Mechanism for software to get an idea on error pattern development over a period of usage. NAND controller will trigger interrupt if the current main page read transfer resulted in correctable errors reached this trigger value for Reed-Solomon selection. For example, of ECC_ERROR interrupt for t=4, with ERR_TRIG_VAL=3 could imply only one of the following. a) If DEC_FAIL = 1, one of the sub-page decode returned failure because no. of symbol errors are more than 4. b) If DEC_FAIL = 0, one of the sub-page decode returned 3 correctable errors. 0 = CORR_ERRS_0 : Reports for every single error, equivalent to ECC_ERROR interrupt without HW_ERR_CORRECTION feature. 1 = CORR_ERRS_1 2 = CORR_ERRS_2 3 = CORR_ERRS_3 4 = CORR_ERRS_4 5 = CORR_ERRS_5 6 = CORR_ERRS_6 7 = CORR_ERRS_7 8 = CORR_ERRS_8 9 = CORR_ERRS_9 10 = CORR_ERRS_10 11 = CORR_ERRS_11 12 = CORR_ERRS_12 13 = CORR_ERRS_13 14 = CORR_ERRS_14 15 = CORR_ERRS_15
15	0x0	IE_RBSY7: Flash7 RBSY line High interrupt enable 0 = DISABLE 1 = ENABLE
14	0x0	IE_RBSY6: Flash6 RBSY line High interrupt enable 0 = DISABLE 1 = ENABLE
13	0x0	IE_RBSY5: Flash5 RBSY line High interrupt enable 0 = DISABLE 1 = ENABLE
12	0x0	IE_RBSY4: Flash4 RBSY line High interrupt enable 0 = DISABLE 1 = ENABLE
11	0x0	IE_RBSY3: Flash3 RBSY line High interrupt enable 0 = DISABLE 1 = ENABLE
10	0x0	IE_RBSY2: Flash2 RBSY line High interrupt enable 0 = DISABLE 1 = ENABLE
9	0x0	IE_RBSY1: Flash1 RBSY line High interrupt enable 0 = DISABLE 1 = ENABLE
8	0x0	IE_RBSY0: Flash0 RBSY line High interrupt enable 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
7	0x0	IE_UND: FIFO underrun interrupt enable 0 = DISABLE 1 = ENABLE
6	0x0	IE_OVR: IFO overrun interrupt enable 0 = DISABLE 1 = ENABLE
5	0x0	IE_CMD_DONE: Command operations done interrupt enable 0 = DISABLE 1 = ENABLE
4	0x0	IE_ECC_ERR: ECC error interrup enable. Please refer to IS_ECC_ERR above for interrupt event details 0 = DISABLE 1 = ENABLE
3	0x0	IE_LL_DONE: Command queue execution completion interrupt enable 0 = DISABLE 1 = ENABLE
2	0x0	IE_LL_ERR: Flash errors in Command queue execution error interrupt enable 0 = DISABLE 1 = ENABLE
0	0x0	GIE: 0 = Global Interrupt mask enable. Masks all of the interrupts, and interrupt to signal to cpu is disabled. 0 = DISABLE 1 = ENABLE

16.4.5 NAND_CONFIG_0

Combination of NAND_COMMAND register determines the sequence of operations to be carried on. ECC selection and strength etc are selectable from this register.

NAND Configuration Register

Offset: 010h | Read/Write: R/W | Reset: 0b00010x00000000110000000000000000

Bit	Reset	Description
31	0x0	HW_ECC: Hardware Error detection enable for Main page read data 0 = DISABLE 1 = ENABLE
30	0x0	ECC_SEL: Hardware Error detection/correction algorithm selection 0 = HAMMING 1 = RS
29	0x0	HW_ERR_CORRECTION: Enable Auto hardware error correction. Emulates software behavior of reading the error vectors from system buffer as pointed in the error vector address register, applies correction and updates the memory word with corrected data. This is done on page basis as soon as the decode information is available as the flash read is placed in memory. 0 = DISABLE 1 = ENABLE
28	0x1	PIPELINE_EN: Pipelines of decode operation enable. Enable next page flash READ data transfer even before current page ECC Decode is completed. If disabled, new page READ is started only after the previous page flash read, ECC decode(detection) are completed. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
27	0x0	ECC_EN_TAG: Hardware Error detection enable for Spare read data 0 = DISABLE 1 = ENABLE
25:24	0x0	TVALUE: Hardware Error correction algorithm strength selection for RS ECC selection. 0 = TVAL4 : (t=4) 4 bit error correction per each 512 bytes of data 1 = TVAL6 : (t=6) 6 bit error correction per each 512 bytes of data 2 = TVAL8 : (t=8) 8 bit error correction per each 512 bytes of data 3 = TVAL_RSVD
23	0x0	SKIP_SPARE: Skip spare feature enable. SKIP_SPAE_SEL below indicates how many bytes in spare area of flash to be skipped over either for reading/writing. 0 = DISABLE 1 = ENABLE
22	0x0	COM_BSY: Combined RBSY selection enabled. 0 = DISABLE: RBSY0 is from Flash card 0 1 = ENABLE: RBSY0 seen by hardware is wired AND of all flash cards connected
21	0x0	BUS_WIDTH: Flash read/write databus width 0 = BUS_WIDTH_8 : selection Databus width 8-bit 1 = BUS_WIDTH_16 : selection Databus width 16-bit
20	0x0	LPDDR1_MODE: LPDDR1 mode of pin ordering enable. Pin ordering to be package friendly with LPDDR1 0 = DISABLE : Standard mode of pin ordering 1 = ENABLE : Pin ordering compatible with LP-DDR1 mcp
19	0x0	EDO_MODE: EDO mode of flash read enable. 0 = DISABLE : Data sampled on posedge of REN 1 = ENABLE : Data sampled on completion of read cycle time
18:16	0x3	PAGE_SIZE_SEL: Page size selection - depends on Flash memory organization used. 0 = PAGE_SIZE_256 1 = PAGE_SIZE_512 2 = PAGE_SIZE_1024 3 = PAGE_SIZE_2048 4 = PAGE_SIZE_4096 5 = PAGE_SIZE_RSVD1 6 = PAGE_SIZE_RSVD2 7 = PAGE_SIZE_RSVD3
15:14	0x0	SKIP_SPARE_SEL: Size in granularity of 4 bytes to skipped for spare access 0 = SKIP_SPARE_SIZE_4 1 = SKIP_SPARE_SIZE_8 2 = SKIP_SPARE_SIZE_12 3 = SKIP_SPARE_SIZE_16
13	0x0	DEBUG_MODE: Debug mode selection for hardware debug
12:9	0x0	DEBUG_SEL: Debug selection for hardware debug
8:0	0x0	TAG_BYTE_SIZE: Block Size in bytes for TAG data from spare area of flash. This is used for specifying the size of the TAG Block data bytes to be move/from to spare area. Used when B_VALID is true. Specified in Bytes (n-1 encoding)

16.4.6 NAND_TIMING_0

Control the flash interface signals for asynchronous timings

where n - value programmed in the respective field of timing register.

NAND Timing Register

Offset: 014h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:28	0x0	TRP_RESP_CNT: Read pulse width(RE Low time)timing for status read cycles Generated timing = (n+1) * NAND_CLK_PERIOD ns GUIDELINE: for tRP_RESP/tRP timing non-EDO mode: Max(tRP, tREA) timing + 6ns (round trip delay) EDO mode: tRP timing from flash datasheet Notes: (1)"round trip delay" to account for - REN out PAD delay + REN out board delay + DATA driven OUT from flash to chip input + DATA INPUT pad delay. (2)For EDO modes - since controller latches data without regard to `nRE' (REN) posedge tREA, round trip delay factors need not be considered.
27:24	0x0	TWB_CNT: WE High to RBSY low asserted (by flash) timing Generated timing = (n+1) * NAND_CLK_PERIOD ns. GUIDELINE: Refer to tWB timing from flash datasheet
23:20	0x0	TCR_TAR_TRR_CNT: RBSY High to RE low timing Generated timing = (n+3) * NAND_CLK_PERIOD ns. GUIDELINE: Program Max(tCR, tAR, tRR) timings from flash data sheet
19:16	0x0	TWHR_CNT: WE High to RE Low timing - Status Read Cycles Generated timing = (n+1) * NAND_CLK_PERIOD ns. GUIDELINE: Refer to tWHR timing from flash data sheet
15:14	0x0	TCS_CNT: CLE/ALE Setup/Hold time. Generated timing = (n+2) * NAND_CLK_PERIOD ns. GUIDELINE: Program Max(tCS, tCH, tALS, tALH) timings from flash datasheet This timing is met timing requirements. 1. from CE Low -> WE posedge of CLE/ALE. 2. from WE posedge of CLE to-> WE posedge of ALE. Please note that this amounts to, 1(above case): (n+2)*NAND_CLK_PERIOD + tWP(of following CLE/ALE cycle) 2(above case): tWH(hold time of current CLE cycle) + (n+2).NAND_CLK_PERIOD of CLE cycle (n+2)*NAND_CLK_PERIOD of ALE cycle + tWP(pusle width of ALE cycle).
13:12	0x0	TWH_CNT: Write pulse HOLD time Generated timing = (n+1) * NAND_CLK_PERIOD ns. GUIDELINE: Refer to tWH timing from flash datasheet

Bit	Reset	Description
11:8	0x0	TWP_CNT: Write pulse width time Generated timing = (n+1) * NAND_CLK_PERIOD ns. GUIDELINE: Refer to tWP timing from flash datasheet
5:4	0x0	TRH_CNT: Read pulse HOLD time Generated timing = (n+1) * NAND_CLK_PERIOD ns. GUIDELINE: Refer to tRH timing from flash datasheet
3:0	0x0	TRP_CNT: Read pulse width(RE Low time)timing for Data read cycles Generated timing = (n+1) * NAND_CLKS, where n - value programmed in the tRP_RESP_CNT field of timing register. GUIDELINE: tRP_RESP/tRP timing register programming non-EDO mode: Max(tRP, tREA) timing + 6ns (round trip delay) EDO mode: tRP timing Notes: (1) "round trip delay" to account for - REN out PAD delay + REN out board delay + DATA driven OUT from flash to chip input + DATA INPUT pad delay. (2) For EDO modes - since controller latches data without regard to `nRE' (REN) posedge tREA, round trip delay factors need not be considered.

16.4.7 NAND_RESP_0

Staging register meant for PIO mode of transfer Contents should be written by CPU access to this register for write transfer. Contents are readable by CPU access to this register after read transfer.

NAND Response Register

Offset: 018h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:24	0x0	BYTE3: Write/Response data byte 3 (MSB)
23:16	0x0	BYTE2: Write/Response data byte 2
15:8	0x0	BYTE1: Write/Response data byte 1
7:0	0x0	BYTE0: Write/Response data byte 0 (LSB)

16.4.8 NAND_TIMING2_0

Controls the flash interface signals asynchronous timings.

NAND Timing2 Register

Offset: 01ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
3:0	0x0	TADL_CNT: WE posedge of address cycle to WE posedge of data cycle Generated timing = (n+3) * NAND_CLK_PERIOD ns. GUIDELINE: Refer to tADL timing from flash datasheet Please note that timing generated from controller is for the duration from ALE low to WP low. In the convention of flash vendor tADL timing this amounts to = (n+3)*NAND_CLK_PERIOD + tWH(previous address cycle) + tWP(following data cycle).

16.4.9 NAND_CMD_REG1_0

Command cycle generation use these during COMMAND1 time:

- CLE cycle1 command value setting

NAND Command1 Register

Offset: 020h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:24	0x0	CMD_BYTE3: Command byte 3(MSB)
23:16	0x0	CMD_BYTE2: Command byte 2
15:8	0x0	CMD_BYTE1: Command byte 1
7:0	0x0	CMD_BYTE0: Command byte 0(LSB)

16.4.10 NAND_CMD_REG2_0

Command cycle generation use these during COMMAND2 time:

- CLE cycle2 command value setting

NAND Command2 Register

Offset: 024h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:24	0x0	CMD_BYTE3: Command byte 3(MSB)
23:16	0x0	CMD_BYTE2: Command byte 2
15:8	0x0	CMD_BYTE1: Command byte 1
7:0	0x0	CMD_BYTE0: Command byte 0(LSB)

16.4.11 NAND_ADDR_REG1_0

Address cycle generation uses these bytes:

- ALE cycle value setting

NAND Address1 Register

Offset: 028h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:24	0x0	ADDR_BYTE3: Address byte 3
23:16	0x0	ADDR_BYTE2: Address byte 2
15:8	0x0	ADDR_BYTE1: Address byte 1
7:0	0x0	ADDR_BYTE0: Address byte 0 (LSB)

16.4.12 NAND_ADDR_REG2_0

Address cycle generation uses these bytes:

- ALE cycle value setting

NAND Addr2 Register

Offset: 02ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:24	0x0	ADDR_BYTE7: Address byte 3
23:16	0x0	ADDR_BYTE6: Address byte 2
15:8	0x0	ADDR_BYTE5: Address byte 1
7:0	0x0	ADDR_BYTE4: Address byte 0 (LSB)

16.4.13 NAND_DMA_MST_CTRL_0

NAND controller DMA interface is configured with this register. Number of page transfers DMA can be initiated only ONCE.

Maximum number of pages transferrable directly depends on the maximum DMA length of 64K bytes.

NAND DMA Master Control Register

Offset: 030h | Read/Write: R/W | Reset: 0b00100100xxx0xxxxxxxxxxxxxxxxxx00

Bit	Reset	Description
31	0x0	DMA_GO: NAND DMA interface enable for data transfers. Hardware clears when programmed length of data transfer is completed. 0 = DISABLE 1 = ENABLE
30	0x0	DIR: DMA data transfer direction. 0 = DMA_RD : 1 = DMA_WR : Read from system and write to flash

Bit	Reset	Description
29	0x1	DMA_PERF_EN: DMA performance feature enable. As soon as the Error vectors equal to BURST SIZE programmed received DMA suspends current data transfers and moves to Error vector transfer and waits till that page decode is completed. Potentially if Error vectors are received around each 512 sub-page boundary this could cause stall of next page READ data transfers causing performance degradation. To take advantage of PIPELINE_EN ECC decoder pipeline capability this should be enabled. 0 = DISABLE : 1 = ENABLE
28	0x0	IE_DMA_DONE: DMA transfer completion interrupt enable 0 = DISABLE 1 = ENABLE
27	0x0	REUSE_BUFFER: Reuse buffer for storing error vectors. 0 = DISABLE, increments the Error Vector destination address continuously till the total DMA transfer size is done 1 = ENABLE, same buffer is used for storing the error vectors for every page decoded
26:24	0x4	BURST_SIZE: DMA burst size selection 0 = BURST_RSVD1 1 = BURST_RSVD2 2 = BURST_1WORDS 3 = BURST_4WORDS 4 = BURST_8WORDS 5 = BURST_16WORDS 6 = BURST_RSVD3 7 = BURST_RSVD4
20	0x0	IS_DMA_DONE: 1 = DMA transfer completed interrupt status. This is set ONLY when not running in COMMAND QUEUE MODE
2	0x0	DMA_EN_A: Enable DMA transfer for Data (A) 0 = DISABLE 1 = ENABLE
1	0x0	DMA_EN_B: Enable DMA transfer for TAG/Spare (B) 0 = DISABLE 1 = ENABLE

16.4.14 NAND_DMA_CFG_A_0

DMA transfer length meant for main area in flash.

NAND DMA Configuration Register for main area

Offset: 034h | Read/Write: R/W | Reset: 0b0000000000000000

Bit	Reset	Description
15:0	0x0	DMA_BLOCK_SIZE_A: DMA Data Block size in Bytes(N-1) value

16.4.15 NAND_DMA_CFG_B_0

DMA transfer length meant for spare area in flash.

NAND DMA Configuration Register for spare area

Offset: 038h | Read/Write: R/W | Reset: 0b0000000000000000

Bit	Reset	Description
15:0	0x0	DMA_BLOCK_SIZE_B: DMA TAG Block size in Bytes(N-1) value

16.4.16 NAND_FIFO_CTRL_0

Indicates FIFO status useful for debug. Also each of these FIFO buffers can be flushed out to come out of abnormal conditions (e.g., controller hang).

NAND FIFO Control/Status Register

Offset: 03ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxx0000

Bit	R/W	Reset	Description
15	RO	X	LL_BUF_EMPTY: 1 = Indicates Command queue FIFO Empty
14	RO	X	LL_BUF_FULL: 1 = Indicates Command queue FIFO Full
13	RO	X	FIFO_A_EMPTY: 1 = Indicates Data FIFO Empty
12	RO	X	FIFO_A_FULL: 1 = Indicates Data FIFO Full
11	RO	X	FIFO_B_EMPTY: 1 = Indicates TAG FIFO Empty
10	RO	X	FIFO_B_FULL: 1 = Indicates TAG FIFO Full
9	RO	X	FIFO_C_EMPTY: 1 = Indicates ECC FIFO Empty
8	RO	X	FIFO_C_FULL: 1 = Indicates ECC FIFO Full
3	RW	0x0	LL_BUF_CLR: field set to "CLEAR_ALL_FIFO" flushes all the buffers(i.e.,LL_BUF,FIFO_A,FIFO_B,FIFO_C) 0 = CLEAR_NO_FIFO 1 = CLEAR_ALL_FIFO
2	RW	0x0	FIFO_A_CLR: Flush the DATA FIFO contents
1	RW	0x0	FIFO_B_CLR: Flush the TAG FIFO contents
0	RW	0x0	FIFO_C_CLR: Flush the ECC FIFO contents

16.4.17 NAND_DATA_BLOCK_PTR_0

NAND Data Block Address Register

Offset: 040h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	DMA_DATA_BLOCK_PTR: DMA data block source/destination address pointer

16.4.18 NAND_TAG_PTR_0

NAND TAG Block Address Register

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

Bit	Reset	Description
31:2	0x0	DMA_TAG_PTR: DMA TAG block source/destination address pointer

16.4.19 NAND_ECC_PTR_0

NAND ECC Block Address Register

Offset: 048h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	DMA_ECC_PTR: DMA Error vector destination address pointer

16.4.20 NAND_DEC_STATUS_0

NAND Decode Status Register

Offset: 04ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:24	X	ERR_PAGE_NUMBER: Indicates the reference to the PAGE for Error Correction to be applied. Valid when IS_ECC_ERROR is generated
23:16	X	ERR_COUNT: No. of Errors occurred in main block READ data plus TAG read data when corresponding features are enabled.
15:8	X	SUB_PAGE_FAIL: Indicates sub-page decode failure within a page size. When decode failure is observed software can use to figure out which sub-page (512 byte) decode failure. for ex: of 2K page size selection, bit 0 - first sub-page bit 1 - second sub-page bit 2 - third sub-page bit 3 - fourth sub-page and so on as applicable
1	X	A_ECC_FAIL: 0 = Main block data decode without decode fail
0	X	B_ECC_FAIL: 0 = Tag block data decode without decode fail

16.4.21 NAND_HWSTATUS_CMD_0

NAND Hardware Status Command Register

Offset: 050h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7:0	0x0	HWSTATUS_CMD: Command byte value used for READ STATUS commands when automatic hardware RBSY_CHK or RD_STATUS_CHK are enabled.

16.4.22 NAND_HWSTATUS_MASK_0

NAND Read Status Mask Register

Offset: 054h | Read/Write: R/W | Reset: 0b11111111111000000100000001000000

Bit	Reset	Description
31:24	0xff	RDSTATUS_MASK: 8 bit Mask value to extract the correct bit fields from READ STATUS information for RD_STATUS_CHK
23:16	0xe0	RDSTATUS_EXP_VAL: 8 bit expected RD STATUS VALUE for RD_STATUS_CHK
15:8	0x40	RBSY_MASK: 8 bit Mask value to extract the correct bit fields from READ STATUS information for RBSY_CHK
7:0	0x40	RBSY_EXP_VAL: 8 bit expected RD STATUS VALUE for RBSY_CHK

16.4.23 NAND_LL_CONFIG_0

NAND Command Queue Configuration Register

Offset: 058h | Read/Write: R/W | Reset: 0b0xxxxxxxxxxx1100xxxx000000000000

Bit	Reset	Description
31	0x0	LL_START: Command Queue feature enable. Hardware clears when command queue data and flash operations are completed. 0 = DISABLE 1 = ENABLE
19	0x1	WORD_CNT_STATUS_EN: Enable word count status update in LL_STATUS register 0 = DISABLE 1 = ENABLE
18:16	0x4	BURST_SIZE: DMA burst size selection for Command Queue data requests 0 = BURST_RSVD1 1 = BURST_RSVD2 2 = BURST_1WORDS 3 = BURST_4WORDS 4 = BURST_8WORDS 5 = BURST_16WORDS 6 = BURST_RSVD3 7 = BURST_RSVD4
11:0	0x0	LL_LENGTH: Command queue up word length programmed is parsed and `START is done when the execution is complete. However when errors are detected for any case of the flash operation failure command queue execution is aborted immediately before this length.

16.4.24 NAND_LL_PTR_0

NAND Command Queue Address Register

Offset: 05ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	LL_PTR: Command queue data pointer Register

16.4.25 NAND_LL_STATUS_0

NAND Command Queue Status Register

Offset: 060h | Read/Write: R/W | Reset: 0bxxxxxxxx00xxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31:24	RO	X	LL_PKT_ID: Command queue PACKET ID completed at this time. Software has write access to this bit field position so that any time software can clear this field. Also NAND controller reset will reset this status.
23	RW	0x0	IS_LL_DONE: Interrupt status of LL_DONE
22	RW	0x0	IS_LL_ERR: Interrupt status of LL_ERR
19:16	RO	X	LL_LENGTH_LAST_PKT: Command queue Word length of last packet executed in the queue. Please note that WORD_CNT_STATUS_EN in LL_CONFIG should be enabled for this status update
11:0	RO	X	LL_LENGTH_DONE: Command queue Word length(32-bit) completed till this time. Please note that WORD_CNT_STATUS_EN in LL_CONFIG should be enabled for this status update

16.4.26 NAND_LOCK_CONTROL_0

Flash erase operation for selected [7:0] apertures are discarded.

NAND Memory Lock Control Register

Offset: 064h | Read/Write: R/W | Reset: 0b000000000

Bit	Reset	Description
8	0x0	IE_LOCK_ERR: Interrupt enable on memory range match. 0 = DISABLE 1 = ENABLE
7	0x0	LOCK_APER_EN7: Enable lock feature for selected apertures 7 Can be set only once register field, hardware reset OR controller reset ONLY can disable this feature. LOCK_APER_START7, LOCK_APER_END7, LOCK_APER_CHIPID7 can't be programmed once this SET 0 = DISABLE 1 = ENABLE
6	0x0	LOCK_APER_EN6: Enable lock feature for selected apertures 6 Can be set only once register field, hardware reset OR controller reset ONLY can disable this feature. LOCK_APER_START6, LOCK_APER_END6, LOCK_APER_CHIPID6 can't be programmed once this SET 0 = DISABLE 1 = ENABLE
5	0x0	LOCK_APER_EN5: Enable lock feature for selected apertures 5 Can be set only once register field, hardware reset OR controller reset ONLY can disable this feature. LOCK_APER_START5, LOCK_APER_END5, LOCK_APER_CHIPID5 can't be programmed once this SET 0 = DISABLE 1 = ENABLE
4	0x0	LOCK_APER_EN4: Enable lock feature for selected apertures 4 Can be set only once register field, hardware reset OR controller reset ONLY can disable this feature. LOCK_APER_START4, LOCK_APER_END4, LOCK_APER_CHIPID4 can't be programmed once this SET 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
3	0x0	LOCK_APER_EN3: Enable lock feature for selected apertures 3 Can be set only once register field, hardware reset OR controller reset ONLY can disable this feature. LOCK_APER_START3, LOCK_APER_END3, LOCK_APER_CHIPID3 can't be programmed once this SET 0 = DISABLE 1 = ENABLE
2	0x0	LOCK_APER_EN2: Enable lock feature for selected apertures 2 Can be set only once register field, hardware reset OR controller reset ONLY can disable this feature. LOCK_APER_START2, LOCK_APER_END2, LOCK_APER_CHIPID2 can't be programmed once this SET 0 = DISABLE 1 = ENABLE
1	0x0	LOCK_APER_EN1: Enable lock feature for selected apertures 1 Can be set only once register field, hardware reset OR controller reset ONLY can disable this feature. LOCK_APER_START1, LOCK_APER_END1, LOCK_APER_CHIPID1 can't be programmed once this SET 0 = DISABLE 1 = ENABLE
0	0x0	LOCK_APER_EN0: Enable lock feature for selected apertures 0 Can be set only once register field, hardware reset OR controller reset ONLY can disable this feature. LOCK_APER_START0, LOCK_APER_END0, LOCK_APER_CHIPID0 can't be programmed once this SET 0 = DISABLE 1 = ENABLE

16.4.27 NAND_LOCK_STATUS_0

Status indicating which aperture matched.

Offset: 068h | Read/Write: R/W | Reset: 0b000000000

Bit	R/W	Reset	Description
8	RW	0x0	IS_LOCK_ERR: 1 = Memory protection error detected check LOCK_STATUS register to identify which aperture matched.
7	RO	0x0	LOCK_APER_STATUS7: Respective bit is set by hardware on protection range detection. Write 1 to clear IS.LOCK_ERR will clear this status information
6	RO	0x0	LOCK_APER_STATUS6: Respective bit is set by hardware on protection range detection. Write 1 to clear IS.LOCK_ERR will clear this status information
5	RO	0x0	LOCK_APER_STATUS5: Respective bit is set by hardware on protection range detection. Write 1 to clear IS.LOCK_ERR will clear this status information
4	RO	0x0	LOCK_APER_STATUS4: Respective bit is set by hardware on protection range detection. Write 1 to clear IS.LOCK_ERR will clear this status information
3	RO	0x0	LOCK_APER_STATUS3: Respective bit is set by hardware on protection range detection. Write 1 to clear IS.LOCK_ERR will clear this status information
2	RO	0x0	LOCK_APER_STATUS2: Respective bit is set by hardware on protection range detection. Write 1 to clear IS.LOCK_ERR will clear this status information
1	RO	0x0	LOCK_APER_STATUS1: Respective bit is set by hardware on protection range detection. Write 1 to clear IS.LOCK_ERR will clear this status information
0	RO	0x0	LOCK_APER_STATUS0: Respective bit is set by hardware on protection range detection. Write 1 to clear IS.LOCK_ERR will clear this status information

16.4.28 NAND_LOCK_APER_START0_0

Program this in the format of "Row Address" NAND flash memory for protection.

NAND Memory Lock Aperture0 Start Address Register

Offset: 06ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	ADDR

16.4.29 NAND_LOCK_APER_START1_0

Program this in the format of "Row Address" NAND flash memory for protection.

NAND Memory Lock Aperture1 Start Address Register

Offset: 070h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	ADDR

16.4.30 NAND_LOCK_APER_START2_0

Program this in the format of "Row Address" NAND flash memory for protection.

NAND Memory Lock Aperture2 Start Address Register

Offset: 074h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	ADDR

16.4.31 NAND_LOCK_APER_START3_0

Program this in the format of "Row Address" NAND flash memory for protection.

NAND Memory Lock Aperture3 Start Address Register

Offset: 078h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	ADDR

16.4.32 NAND_LOCK_APER_START4_0

Program this in the format of "Row Address" NAND flash memory for protection.

NAND Memory Lock Aperture4 Start Address Register

Offset: 07ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	ADDR

16.4.33 NAND_LOCK_APER_START5_0

Program this in the format of "Row Address" NAND flash memory for protection.

NAND Memory Lock Aperture5 Start Address Register

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

Bit	Reset	Description
31:0	X	ADDR

16.4.34 NAND_LOCK_APER_START6_0

Program this in the format of "Row Address" NAND flash memory for protection.

NAND Memory Lock Aperture6 Start Address Register

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

Bit	Reset	Description
31:0	X	ADDR

16.4.35 NAND_LOCK_APER_START7_0

Program this in the format of "Row Address" NAND flash memory for protection.

NAND Memory Lock Aperture7 Start Address Register

Offset: 088h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	ADDR

16.4.36 NAND_LOCK_APER_END0_0

Program this in the format of "Row Address" NAND flash memory for protection.

NAND Memory Lock Aperture0-7 End Address Register

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

Bit	Reset	Description
31:0	X	ADDR

16.4.37 NAND_LOCK_APER_END1_0

Program this in the format of "Row Address" NAND flash memory for protection.

NAND Memory Lock Aperture0-7 End Address Register

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

Bit	Reset	Description
31:0	X	ADDR

16.4.38 NAND_LOCK_APER_END2_0

Program this in the format of "Row Address" NAND flash memory for protection.

NAND Memory Lock Aperture0-7 End Address Register

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

Bit	Reset	Description
31:0	X	ADDR

16.4.39 NAND_LOCK_APER_END3_0

Program this in the format of "Row Address" NAND flash memory for protection.

NAND Memory Lock Aperture0-7 End Address Register

Offset: 098h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	ADDR

16.4.40 NAND_LOCK_APER_END4_0

Program this in the format of "Row Address" NAND flash memory for protection.

NAND Memory Lock Aperture0-7 End Address Register

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

Bit	Reset	Description

Bit	Reset	Description
31:0	X	ADDR

16.4.41 NAND_LOCK_APER_END5_0

Program this in the format of "Row Address" NAND flash memory for protection.

NAND Memory Lock Aperture0-7 End Address Register

Offset: 0a0h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	ADDR

16.4.42 NAND_LOCK_APER_END6_0

Program this in the format of "Row Address" NAND flash memory for protection.

NAND Memory Lock Aperture0-7 End Address Register

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

Bit	Reset	Description
31:0	X	ADDR

16.4.43 NAND_LOCK_APER_END7_0

Program this in the format of "Row Address" NAND flash memory for protection.

NAND Memory Lock Aperture0-7 End Address Register

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

Bit	Reset	Description
31:0	X	ADDR

16.4.44 NAND_LOCK_APER_CHIPID0_0

NAND Memory Lock Aperture0-7 Chip Select Register

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

Bit	Reset	Description
7	X	CS7: Memory Protection of aperture[0-7] valid for Chip select7 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select7 1 = ENABLE
6	X	CS6: Memory Protection of aperture[0-7] valid for Chip select6 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select6 1 = ENABLE
5	X	CS5: Memory Protection of aperture[0-7] valid for Chip select5 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select5 1 = ENABLE

Bit	Reset	Description
4	X	CS4: Memory Protection of aperture[0-7] valid for Chip select4 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select4 1 = ENABLE
3	X	CS3: Memory Protection of aperture[0-7] valid for Chip select3 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select3 1 = ENABLE
2	X	CS2: Memory Protection of aperture[0-7] valid for Chip select2 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select2 1 = ENABLE
1	X	CS1: Memory Protection of aperture[0-7] valid for Chip select1 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select1 1 = ENABLE
0	X	CS0: Memory Protection of aperture[0-7] valid for Chip select0 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select0 1 = ENABLE

16.4.45 NAND_LOCK_APER_CHIPID1_0

NAND Memory Lock Aperture0-7 Chip Select Register

Offset: 0b0h | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7	X	CS7: Memory Protection of aperture[0-7] valid for Chip select7 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select7 1 = ENABLE
6	X	CS6: Memory Protection of aperture[0-7] valid for Chip select6 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select6 1 = ENABLE
5	X	CS5: Memory Protection of aperture[0-7] valid for Chip select5 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select5 1 = ENABLE
4	X	CS4: Memory Protection of aperture[0-7] valid for Chip select4 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select4 1 = ENABLE
3	X	CS3: Memory Protection of aperture[0-7] valid for Chip select3 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select3 1 = ENABLE
2	X	CS2: Memory Protection of aperture[0-7] valid for Chip select2 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select2 1 = ENABLE
1	X	CS1: Memory Protection of aperture[0-7] valid for Chip select1 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select1 1 = ENABLE
0	X	CS0: Memory Protection of aperture[0-7] valid for Chip select0 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select0 1 = ENABLE

16.4.46 NAND_LOCK_APER_CHIPID2_0

NAND Memory Lock Aperture0-7 Chip Select Register

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

Bit	Reset	Description
7	X	CS7: Memory Protection of aperture[0-7] valid for Chip select7 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select7 1 = ENABLE
6	X	CS6: Memory Protection of aperture[0-7] valid for Chip select6 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select6 1 = ENABLE
5	X	CS5: Memory Protection of aperture[0-7] valid for Chip select5 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select5 1 = ENABLE
4	X	CS4: Memory Protection of aperture[0-7] valid for Chip select4 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select4 1 = ENABLE
3	X	CS3: Memory Protection of aperture[0-7] valid for Chip select3 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select3 1 = ENABLE
2	X	CS2: Memory Protection of aperture[0-7] valid for Chip select2 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select2 1 = ENABLE
1	X	CS1: Memory Protection of aperture[0-7] valid for Chip select1 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select1 1 = ENABLE
0	X	CS0: Memory Protection of aperture[0-7] valid for Chip select0 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select0 1 = ENABLE

16.4.47 NAND_LOCK_APER_CHIPID3_0

NAND Memory Lock Aperture0-7 Chip Select Register

Offset: 0b8h | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7	X	CS7: Memory Protection of aperture[0-7] valid for Chip select7 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select7 1 = ENABLE
6	X	CS6: Memory Protection of aperture[0-7] valid for Chip select6 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select6 1 = ENABLE
5	X	CS5: Memory Protection of aperture[0-7] valid for Chip select5 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select5 1 = ENABLE
4	X	CS4: Memory Protection of aperture[0-7] valid for Chip select4 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select4 1 = ENABLE

Bit	Reset	Description
3	X	CS3: Memory Protection of aperture[0-7] valid for Chip select3 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select3 1 = ENABLE
2	X	CS2: Memory Protection of aperture[0-7] valid for Chip select2 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select2 1 = ENABLE
1	X	CS1: Memory Protection of aperture[0-7] valid for Chip select1 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select1 1 = ENABLE
0	X	CS0: Memory Protection of aperture[0-7] valid for Chip select0 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select0 1 = ENABLE

16.4.48 NAND_LOCK_APER_CHIPID4_0

NAND Memory Lock Aperture0-7 Chip Select Register

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

Bit	Reset	Description
7	X	CS7: Memory Protection of aperture[0-7] valid for Chip select7 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select7 1 = ENABLE
6	X	CS6: Memory Protection of aperture[0-7] valid for Chip select6 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select6 1 = ENABLE
5	X	CS5: Memory Protection of aperture[0-7] valid for Chip select5 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select5 1 = ENABLE
4	X	CS4: Memory Protection of aperture[0-7] valid for Chip select4 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select4 1 = ENABLE
3	X	CS3: Memory Protection of aperture[0-7] valid for Chip select3 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select3 1 = ENABLE
2	X	CS2: Memory Protection of aperture[0-7] valid for Chip select2 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select2 1 = ENABLE
1	X	CS1: Memory Protection of aperture[0-7] valid for Chip select1 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select1 1 = ENABLE
0	X	CS0: Memory Protection of aperture[0-7] valid for Chip select0 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select0 1 = ENABLE

16.4.49 NAND_LOCK_APER_CHIPID5_0

NAND Memory Lock Aperture0-7 Chip Select Register

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

Bit	Reset	Description
7	X	CS7: Memory Protection of aperture[0-7] valid for Chip select7 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select7 1 = ENABLE
6	X	CS6: Memory Protection of aperture[0-7] valid for Chip select6 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select6 1 = ENABLE
5	X	CS5: Memory Protection of aperture[0-7] valid for Chip select5 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select5 1 = ENABLE
4	X	CS4: Memory Protection of aperture[0-7] valid for Chip select4 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select4 1 = ENABLE
3	X	CS3: Memory Protection of aperture[0-7] valid for Chip select3 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select3 1 = ENABLE
2	X	CS2: Memory Protection of aperture[0-7] valid for Chip select2 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select2 1 = ENABLE
1	X	CS1: Memory Protection of aperture[0-7] valid for Chip select1 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select1 1 = ENABLE
0	X	CS0: Memory Protection of aperture[0-7] valid for Chip select0 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select0 1 = ENABLE

16.4.50 NAND_LOCK_APER_CHIPID6_0

NAND Memory Lock Aperture0-7 Chip Select Register

Offset: 0c4h | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7	X	CS7: Memory Protection of aperture[0-7] valid for Chip select7 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select7 1 = ENABLE
6	X	CS6: Memory Protection of aperture[0-7] valid for Chip select6 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select6 1 = ENABLE
5	X	CS5: Memory Protection of aperture[0-7] valid for Chip select5 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select5 1 = ENABLE
4	X	CS4: Memory Protection of aperture[0-7] valid for Chip select4 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select4 1 = ENABLE

Bit	Reset	Description
3	X	CS3: Memory Protection of aperture[0-7] valid for Chip select3 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select3 1 = ENABLE
2	X	CS2: Memory Protection of aperture[0-7] valid for Chip select2 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select2 1 = ENABLE
1	X	CS1: Memory Protection of aperture[0-7] valid for Chip select1 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select1 1 = ENABLE
0	X	CS0: Memory Protection of aperture[0-7] valid for Chip select0 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select0 1 = ENABLE

16.4.51 NAND_LOCK_APER_CHIPID7_0

NAND Memory Lock Aperture0-7 Chip Select Register

Offset: 0c8h | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7	X	CS7: Memory Protection of aperture[0-7] valid for Chip select7 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select7 1 = ENABLE
6	X	CS6: Memory Protection of aperture[0-7] valid for Chip select6 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select6 1 = ENABLE
5	X	CS5: Memory Protection of aperture[0-7] valid for Chip select5 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select5 1 = ENABLE
4	X	CS4: Memory Protection of aperture[0-7] valid for Chip select4 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select4 1 = ENABLE
3	X	CS3: Memory Protection of aperture[0-7] valid for Chip select3 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select3 1 = ENABLE
2	X	CS2: Memory Protection of aperture[0-7] valid for Chip select2 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select2 1 = ENABLE
1	X	CS1: Memory Protection of aperture[0-7] valid for Chip select1 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select1 1 = ENABLE
0	X	CS0: Memory Protection of aperture[0-7] valid for Chip select0 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select0 1 = ENABLE

16.4.52 NAND_BCH_CONFIG_0

NAND BCH Error Correction Control Register

Offset: 0cch | Read/Write: R/W | Reset: 0b00xxx0

Bit	Reset	Description
5:4	0x0	BCH_TVALUE: BCH error correction strength selection. 0 = BCH_TVAL4 : 4 single bit random errors per sector 1 = BCH_TVAL8 : 8 single bit random errors per sector 2 = BCH_TVAL14 : 14 single bit random errors per sector 3 = BCH_TVAL16
0	0x0	BCH_ECC: BCH ECC scheme is is enabled 0 = DISABLE 1 = ENABLE

16.4.53 NAND_BCH_DEC_RESULT_0

NAND BCH/RS/Hamming extended decode status information

- CORRFAIL_ERR flag is added to NAND_ISR register to indicate that out of total DMA transfers of specified pages there are correctable OR un-correctable errors. Please check the description field below. If there are no correctable/un-correctable errors this bit will not be SET.
- New register NAND_DEC_RESULT indicates the count of pages resulted in correctable/un-correctable errors.
- New register NAND_DEC_STAT_BUF. Reading this register will fetch out an entry from decode status buffer. Each entry of this buffer relate to page which resulted in either correctable/un-correctable error. Please check the description of this entry described as NAND_DEC_STAT_EXT below. It is up to s/w how this can be used; hopefully this serves for all purposes.

Example

If there are 4 pages resulted in correctable error and 2 pages of un-correctable error out of total DMA transfer of 64 pages, PAGE_COUNT in NAND_DEC_RESULT will indicate as PAGE_COUNT=6. Software should read these 6 entries reading NAND_DEC_STAT_BUF register by CPU access and process it for further action.

Offset: 0d0h | Read/Write: RO | Reset: 0bxxxxxxxxx

Bit	Reset	Description
8	X	CORRFAIL_ERR: 1 = Correctable OR Un-correctable errors occurred in the DMA transfer without regard to HW_ERR_CORRECTION feature is enabled or not. Use extended decode results in NAND_DEC_RESULT and NAND_DEC_STATUS_EXT to figure out further action for block replacement/wear leveling during file system management for s/w.
7:0	X	PAGE_COUNT: No. of pages resulted either in un-correctable or correctable errors

16.4.54 NAND_BCH_DEC_STATUS_BUF_0

Offset: 0d4h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:24	X	FAIL_SEC_FLAG: Sector wise un-correctable error indicator Bit 31 = 1, sector 7 has un-correctable errors Bit 31 = 0, sector 7 has no un-correctable errors ... Bit 24 = 1, sector 0 has un-correctable errors Bit 24 = 0, sector 0 has no un-correctable errors

Bit	Reset	Description
23:16	X	CORR_SEC_FLAG: Sector wise correctable error indicator Bit 23 = 1, sector 7 has correctable errors Bit 23 = 0, sector 7 has no correctable errors ... Bit 16 = 1, sector 0 has correctable errors Bit 16 = 0, sector 0 has no correctable errors
14	X	FAIL_TAG: Spare area error decode resulted in un-correctable errors in case of RS/Hamming ECC selection. For BCH this field is not applicable.
13	X	CORR_TAG: Spare area error decode resulted in correctable errors in case of RS/Hamming ECC selection. For BCH this field is not applicable.
12:8	X	MAX_CORR_CNT: Maximum no. of correctable errors occurred out of all sectors. For example of 2K page, if sector 0 has 2 correctable errors and sector3 has 4 errors MAX_ERR_CNT will reflect as 4
7:0	X	PAGE_NUMBER: Page number which resulted in either correctable/un-correctable errors 0 to 63 indication for 64 pages of DMA transfer.

17.0 GMI CONTROLLER

The Generic Memory Interface (GMI) is a controller which supports a number of memory types, including synchronous NOR, asynchronous NOR, and other flash memories with similar interfaces such as MuxOneNAND.

The GMI Controller is also known as the Sync-NOR (SNOR) Controller, and enables:

- Data transfer between internal and external memory.
- Interface with Synchronous/Asynchronous Flash Memories.
- Ability to boot from MuxOneNAND.

The GMI/SNOR Controller is on the AHB bus, it provides a way to access external NOR-Flash through Master as well as Slave modes. The SNOR controller supports DMA transfer between Flash Memory and System Memory to save on processor cycles. In order to reduce the number of I/O pins, the NOR interface and NAND interface share pins.

SNOR controller can operate in 2 modes:

- Programmed IO (PIO) mode – In this case the CPU/COP operates as master and issues the read/write commands, and the SNOR controller operates as the slave. The SNOR controller handles the transfer of data between the flash memory and CPU registers or COP
- DMA mode – In this case the SNOR controller operates as the master; the SNOR controller handles the transfer of data between the flash memory and system memory or IRAM

17.1 Functional Description

The GMI/SNOR Controller supports the following operation:

- Synchronous and Asynchronous Flash Memories support
- MuxOneNAND support
- AHB Master and Slave mode support
- Booting Option
- Address-Data Mux / Non-Mux Configuration
- Fast Access Modes – Burst (Reads/Writes) and Page Mode (Reads)
- Programmable WAIT states for Asynchronous access
- Programmable Data RDY Configuration (same cycle or next cycle)
- Support 16/32 bits wide Data Bus
- Support for 8 Chip selects: Any combination of 8-SNOR chip selects

17.1.1 GMI Functional Waveforms

Table 57 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 250 and GMI device 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 28 Synchronous Read (GMI Non-Mux Device)

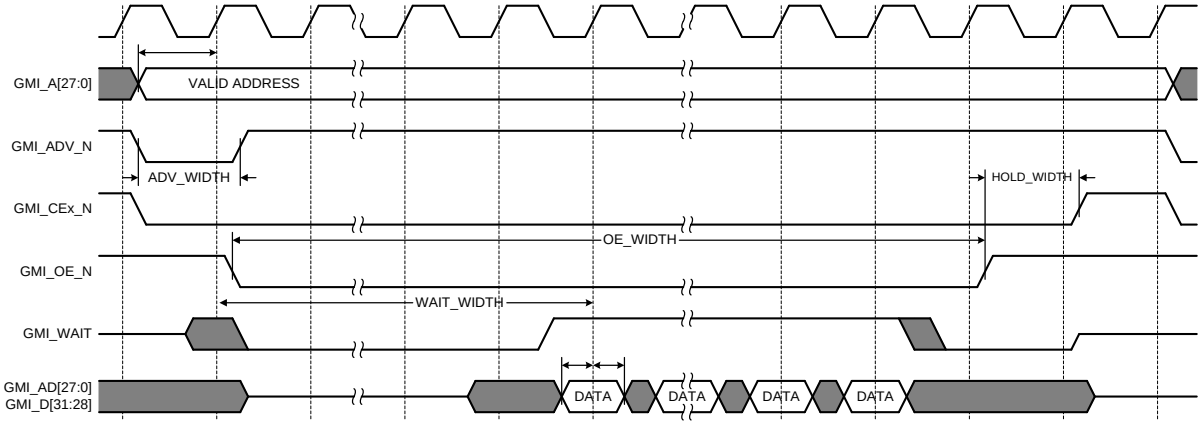


Figure 29 Synchronous Read (GMI Mux-AD 16-bit Data Device)

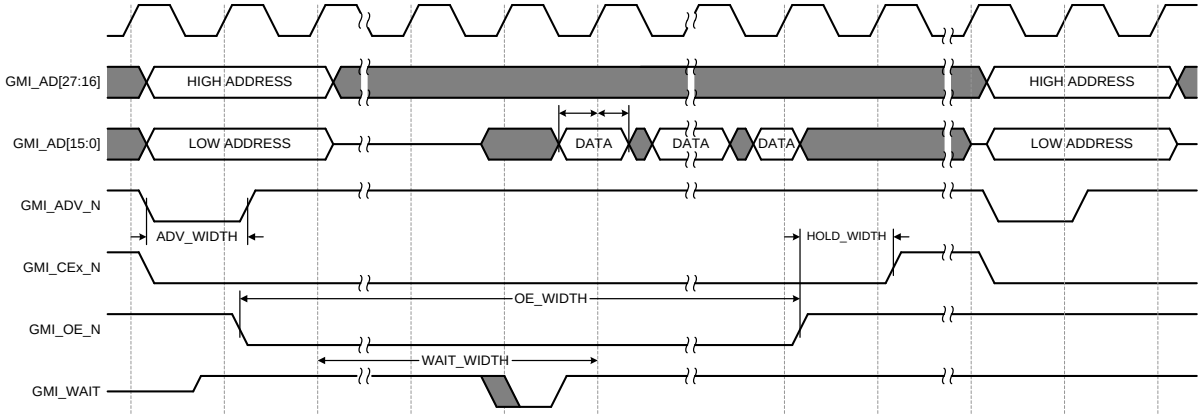


Figure 30 Asynchronous Read (GMI MUX-AD 16-bit Data Device)

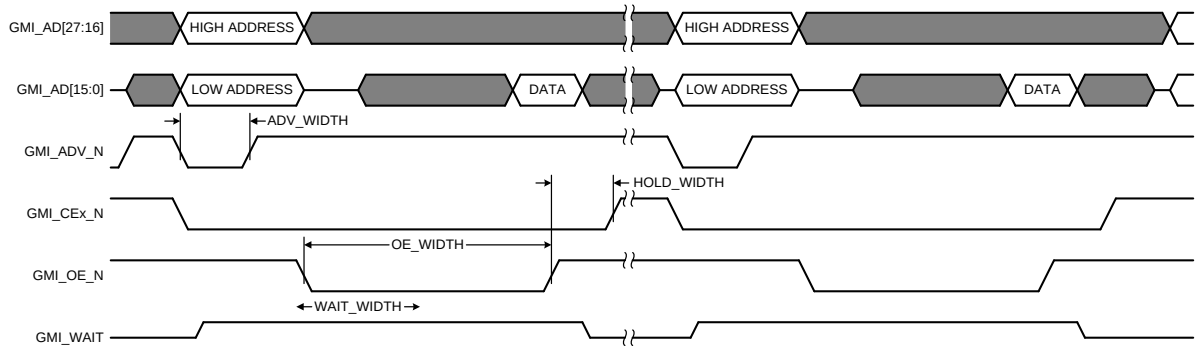
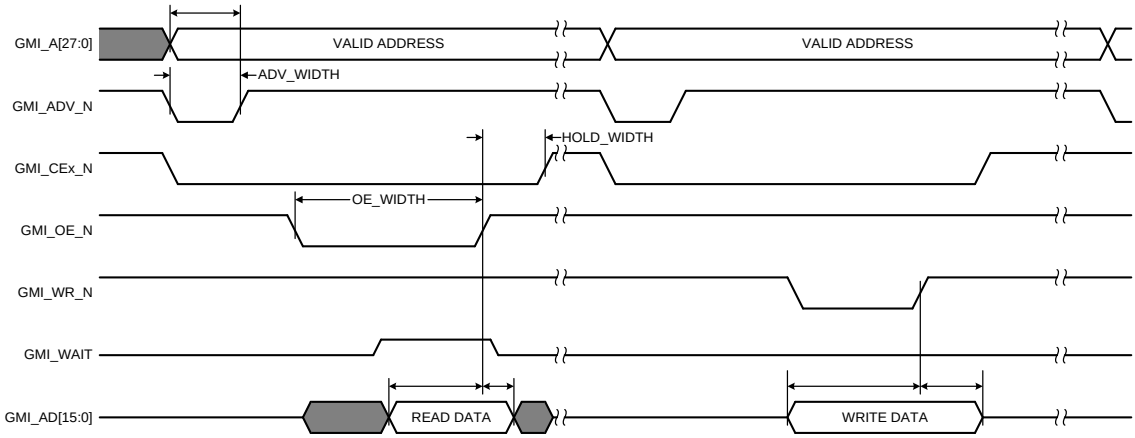


Figure 31 Asynchronous Read Followed by Asynchronous Write (GMI Non-Mux 16-bit Data Device)



17.1.2 GMI Memory Mapping

Table 58 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 59 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

17.2 Programming Guidelines

At power-on reset the GMI/SNOR Controller is active in slave mode. 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:

- SNOR timing registers should be programmed per the device specifications.
- Using the Slave Interface Asynchronous Writes/Reads are permitted.
- Use the device configuration command cycles to enable the different features of the device including Burst Mode.
- 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 “DMA_GO” and “GO_NOR” Bits. Read the register description for more information.
- Enable the “DMA_GO” bit followed by “GO_NOR” bit to start the DMA operation.
- DMA status can be seen by polling the DMA Status Busy Signal or using the transfer done interrupt.

17.3 GMI Registers

17.3.1 SNOR_CONFIG_0

SNOR Configuration Register

Offset: 000h | Read/Write: R/W | Reset: 0b000100x010000xxx0xxxx00000000000

Bit	Reset	Description
31	0x0	GO_NOR: Enable this bit starts the NOR operation. This bit should be written only after the rest of the NOR parameters are programmed. 0 = DISABLE 1 = ENABLE
30	0x0	WORDWIDE_GMI: NOR Device DataBus width Configuration Bit. 0 = NOR16BIT 1 = NOR32BIT
29	0x0	NOR_DEVICE_TYPE: External NOR Memory Type. 0 = SNOR 1 = MUXONENAND (simulation purpose)
28	0x1	MUXMODE_GMI: NOR Device Address-Data Configuration Bit. 0 = AD_NONMUX 1 = AD_MUX
27:26	0x0	BURST_LENGTH: Burst Length Types 00=Continuous Burst, 01=8 Words, 10=16 Words, 11=32 Words. 0 = CNTBRST 1 = BL8WORD 2 = BL16WORD 3 = BL32WORD
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

Bit	Reset	Description
10:8	0x0	PAGE_SIZE: Sets the number of words in a page if page mode is selected. 0 = BRST 1 = PG4WORD 2 = PG8WORD 3 = PG16WORD 4 = RESV4 5 = RESV5 6 = RESV6 7 = RESV7
7	0x0	MST_ENB: Selection bit between Master DMA and Slave Interface. 0 = DISABLE 1 = ENABLE
6:4	0x0	SNOR_SEL: SNOR 8 chip selects combinations. 0 = CS0 1 = CS1 2 = CS2 3 = CS3 4 = CS4 5 = CS5 6 = CS6 7 = CS7
3	0x0	CE_LAST: Indicates if the ADV gets asserted before CE. 0 = DISABLE 1 = RESV
2	0x0	CE_FIRST: Indicates if the CE gets asserted before ADV. 0 = DISABLE 1 = RESV
1:0	0x0	DEVICE_MODE: This field specifies the Mode of Operation for SYNC Memories. 0 = ASYNC 1 = PAGE 2 = BURST 3 = RESV

17.3.2 SNOR_STA_0

This status register has bits that provide the controller status, along with the external interrupts from the SNOR devices. The FIFO status is also provided along with the status on the count of the words transferred through DMA

SNOR Status Register

Offset: 004h | Read/Write: R/W | Reset: 0bxxxx0000xxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	DEVICE_BSY: Indicates that the device status.
27	RW	0x0	DEVICE_INTR_2: Device Interrupt-2 from MuxOneNand Memory.
26	RW	0x0	DEVICE_INTR_1: Device Interrupt-1 from MuxOneNand Memory.
25	RW	0x0	DEVICE_INTR_2_ENB: Device Interrupt-2 Enable Bit.
24	RW	0x0	DEVICE_INTR_1_ENB: Device Interrupt-1 Enable Bit.
23	RO	X	SLV_FIFO_FULL: SLV FIFO full status.

Bit	R/W	Reset	Description
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.

17.3.3 SNOR_NOR_ADDR_PTR_0

Contains the starting address of the NOR access.

SNOR Address Register

Offset: 008h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	SNOR_NOR_ADDR_PTR: Indicates that the NOR controller Address.

17.3.4 SNOR_AHB_ADDR_PTR_0

Contains the starting address of the AHB access.

SNOR AHB Address Register

Offset: 00ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	SNOR_AHB_ADDR_PTR: Indicates that the AHB side Address.

17.3.5 SNOR_TIMING0_0

This register contains the timing parameters used for the page access, along with the timing parameters for the widths of ADV pulse, CE hold, etc.

SNOR Timing0 Register

Offset: 010h | Read/Write: R/W | Reset: 0b0011xxxx0001xxxx0001000100010100

Bit	Reset	Description
31:28	0x3	PAGE_RDY_WIDTH: This represents the number of wait clock cycles from address to 1st data ready. Programming 0 means 1 clock cycle: actual cycle = programmed cycle + 1
23:20	0x1	PAGE_SEQ_WIDTH: Page Sequential width indicates the delay cycle between the intra page Read access. Programming 0 means 1 clock cycle: actual cycle = programmed cycle + 1
15:12	0x1	MUXED_WIDTH: Indicates in number of cycles MUX address/data asserted on the bus. Programming 0 means 1 clock cycle: actual cycle = programmed cycle + 1
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). Programming 0 means 1 clock cycle: actual cycle = programmed cycle + 1

Bit	Reset	Description
7:4	0x1	ADV_WIDTH: Indicates the number of cycles during which ADV stays asserted. Programming 0 means 1 clock cycle: actual cycle = programmed cycle + 1
3:0	0x4	CE_WIDTH: Indicates the number of cycles before CE is asserted. Fixed (minimum 1 cycle).

17.3.6 SNOR_TIMING1_0

This register contains the timing parameters of the WE and OE widths along with the WAIT width.

SNOR Timing1 Register.

Offset: 014h | Read/Write: R/W | Reset: 0b00000001000000010000011

Bit	Reset	Description
23:16	0x1	WR_WIDTH: Write access time. Programming 0 means 1 clock cycle: actual cycle = programmed cycle + 1
15:8	0x1	OE_WIDTH: Read access time. Programming 0 means 1 clock cycle: actual cycle = programmed cycle + 1
7:0	0x3	WAIT_WIDTH: Indicates in cycles the number of wait states before when READY is issued. Programming 0 means 1 clock cycle: actual cycle = programmed cycle + 1

17.3.7 SNOR_MIO_CFG_0

This register contains the MIO configuration bits such as the width of the words, the polarity of the ready along with the three bits to select the 8 different chip selects for MIO.

MIO Configuration Register

Offset: 018h | Read/Write: R/W | Reset: 0b01xxxxx111

Bit	Reset	Description
29	0x0	MIO_WORDWIDE: Indicates the databus size of MIO Memory. 0 = MIO16BIT 1 = MIO32BIT
28	0x1	MIO_RDY_POL: Specifies the polarity of MIO RDY. 0 = RESV 1 = HIGH
22:20	0x7	MIO_SEL: MIO 8 chip selects combinations. 0 = MIO0 1 = MIO1 2 = MIO2 3 = MIO3 4 = MIO4 5 = MIO5 6 = MIO6 7 = MIO7

17.3.8 SNOR_MIO_TIMING0_0

This register contains the timing parameters for the MIO accesses MIO Timing - 0 Register

Offset: 01ch | Read/Write: R/W | Reset: 0b000001xx000010xx000001xx000010

Bit	Reset	Description
29:24	0x1	MIO_A_DED_WR: Minimum number of MIO bus clock cycles between the end of a write access and the start of the following access (write or read) for MIO.
21:16	0x2	MIO_A_WR_TIME: Minimum number of MIO bus clock cycles during a write access that the MIO_RD signal is set low for MIO.
13:8	0x1	MIO_A_DED_RD: Minimum number of MIO bus clock cycles between the end of a read access and the start of the following access (write or read) for MIO.
5:0	0x2	MIO_A_RD_TIME: Minimum number of MIO bus clock cycles during a read access that the MIO_RD signal is set low for MIO.

17.3.9 SNOR_DMA_CFG_0

SNOR DMA Configuration Register

Offset: 020h | Read/Write: R/W | Reset: 0b00000100xxxxxxxx00000000000000

Bit	Reset	Description
31	0x0	DMA_GO: Enabling this bit starts the DMA operation This bit should be written only after the rest of the DMA parameters are programmed. 0 = DISABLE 1 = ENABLE
30	0x0	BSY: Indicates the status of DMA. 0= Idle 1= Busy
29	0x0	DIR: This represents the direction of DMA data Transfer. 0 = NOR2AHB 1 = AHB2NOR
28	0x0	IE_DMA_DONE: Interrupt Enable on DMA transfer completion. 0 = DISABLE 1 = ENABLE
27	0x0	IS_DMA_DONE: Interrupt Status (Write 1 to clear). 0 = DISABLE 1 = ENABLE
26:24	0x4	BURST_SIZE: DMA burst size. 0 = RESV0 1 = RESV1 2 = RESV2 3 = RESV3 4 = BS1WORD 5 = BS4WORD 6 = BS8WORD 7 = RESV7
15:2	0x0	WORD_COUNT: Specifies the number of words that need to be transferred.

17.3.10 SNOR_CS_MUX_CFG_0

SNOR CHIP Select MUX Configuration Register

Offset: 024h | Read/Write: R/W | Reset: 0b111x110x101x100x011x010x001x000

Bit	Reset	Description
30:28	0x7	CS7_MUX: This represents which chip selects goes to which memory. Chip selection between SNOR and MIO Memories.
26:24	0x6	CS6_MUX: This represents which chip selects goes to which memory. Chip selection between SNOR and MIO Memories.
22:20	0x5	CS5_MUX: This represents which chip selects goes to which memory. Chip selection between SNOR and MIO Memories.
18:16	0x4	CS4_MUX: This represents which chip selects goes to which memory. Chip selection between SNOR and MIO Memories.
14:12	0x3	CS3_MUX: This represents which chip selects goes to which memory. Chip selection between SNOR and MIO Memories.
10:8	0x2	CS2_MUX: This represents which chip selects goes to which memory. Chip selection between SNOR and MIO Memories.
6:4	0x1	CS1_MUX: This represents which chip selects goes to which memory. Chip selection between SNOR and MIO Memories.
2:0	0x0	CS0_MUX: This represents which chip selects goes to which memory. Chip selection between SNOR and MIO Memories.

18.0 GPIO CONTROLLER

The Tegra® 2 Processor provides a large number of potential general-purpose I/O (GPIO) pins. The actual number of available GPIO pins depends on the system pin-mux configuration. These pins are grouped as twenty-eight 8-bit Ports. The ports are grouped into 7 GPIO controllers containing four 8-bit ports each. All these port pins are designed with multiplexing logic, so they can be used as either a dedicated Special Function I/O (SFIO) or a GPIO.

Features

- Multiplexed GPIO/SFIO functionality
- Protection against meta-stability for GPIO inputs (back-to-back D flip-flops)
- Interrupts are programmable on each individual GPIO pin
- Interrupt levels also programmable
- All pin configurations are independent of the other pins
- Masked-write registers

> **Note:** Some pin attributes that affect GPIO operation, including Tristate, Pullup/down, drive/schmidt control, are on Pin Group or Pin Config (group) granularity. These controls are outside the GPIO block.

Each port has a set of configuration registers, which are used for:

- GPIO/SFIO selection
- Enabling outputs
- Driving the output pin with a HIGH or LOW
- Receiving the input when in GPIO mode
- Enabling interrupts
- Checking status
- Clearing interrupts

18.1 Functionality

There are 222 (Tegra 250) or 211 (AP20) pins that can function as either multiplexed Special Function I/O (SFIO) pins or General-Purpose I/O (GPIO) pins. In Special function mode, these pins are driven by the pinmux controller, where, each pin is driven by a special function controller selected based on the pinmux option (primary, alternate-1, alternate-2 and alternate-3).

This multiplexing architecture helps to support a much higher number of IOs on a minimal number of pads on the chip. There are additional pins too, that function as SFIOs only, without sharing the GPIO function. GPIO programmability gives more flexibility to the firmware. In addition to this, the GPIO architecture gives Firmware the flexibility of receiving any interrupt from an external requestor on any GPIO pin.

GPIO Pins

Each GPIO controller consists of 4 GPIO ports. Each port has a set of registers capable of controlling 8 GPIO pins. The ports for the first controller are named A, B, C, and D. The ports for the second controller are named E, F, G, and H. To find the offset of a particular register for a particular port, add the port offset from Table 60 to the register's offset. For example, the offset of GPIO Port R's GPIO_OUT register is 0x204 (Port Offset) + 0x20 (Register Offset) = 0x224.

Table 60: GPIO Controllers, Ports, and Pins

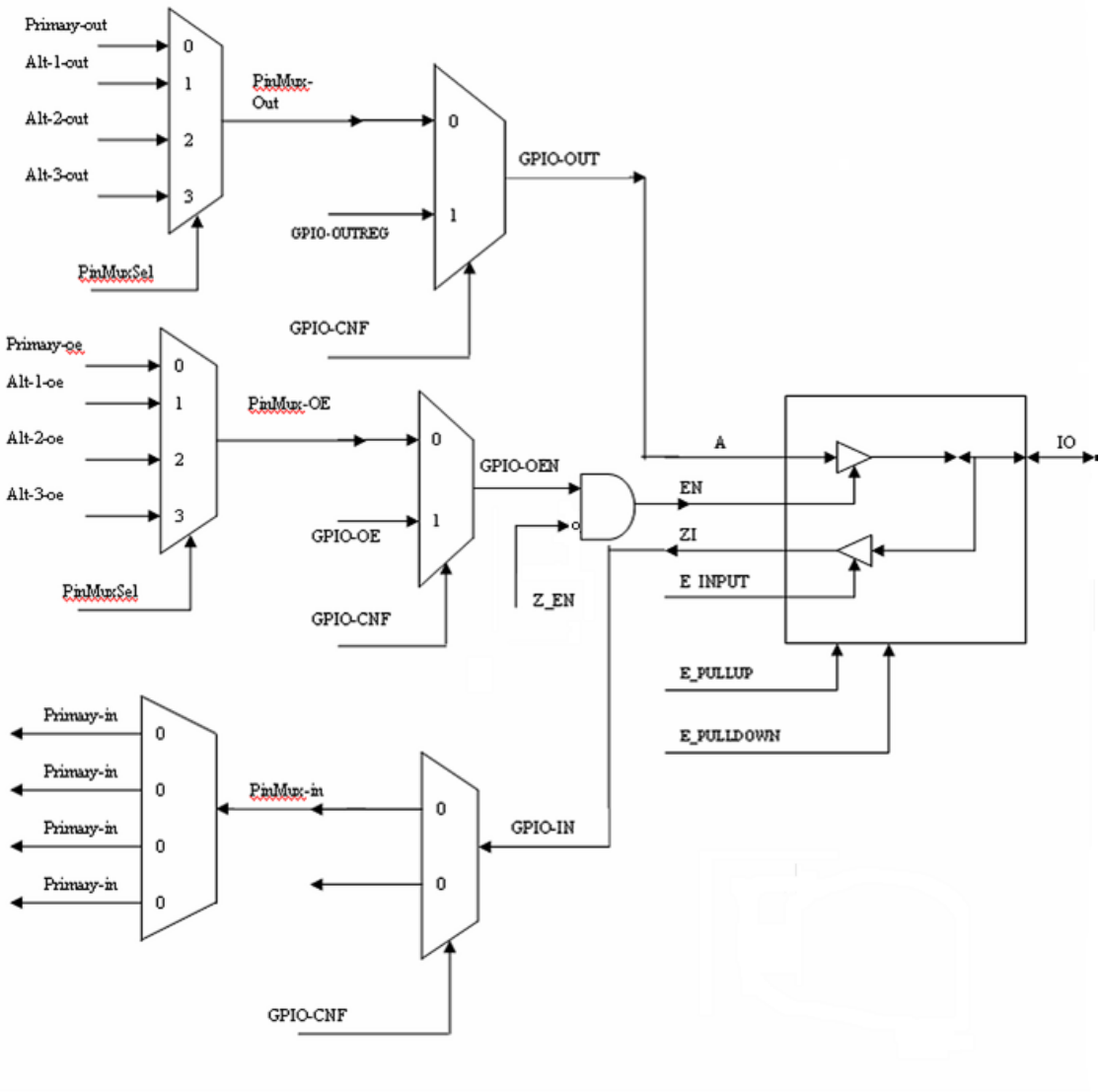
Controller	Port	Port Offset	Pins
GPIO1	Port A	0x000	GPIO_PA0 - GPIO_PA7
	Port B	0x004	GPIO_PB0 - GPIO_PB7
	Port C	0x008	GPIO_PC0 - GPIO_PC7
	Port D	0x00C	GPIO_PD0 - GPIO_PD7
GPIO2	Port E	0x080	GPIO_PE0 - GPIO_PE7
	Port F	0x084	GPIO_PF0 - GPIO_PF7
	Port G	0x088	GPIO_PG0 - GPIO_PG7
	Port H	0x08C	GPIO_PH0 - GPIO_PH7
GPIO3	Port I	0x100	GPIO_PI0 - GPIO_PI7
	Port J	0x104	GPIO_PJ0 - GPIO_PJ7
	Port K	0x108	GPIO_PK0 - GPIO_PK7
	Port L	0x10C	GPIO_PL0 - GPIO_PL7
GPIO4	Port M	0x180	GPIO_PM0 - GPIO_PM7
	Port N	0x184	GPIO_PN0 - GPIO_PN7
	Port O	0x188	GPIO_PO0 - GPIO_PO7
	Port P	0x18C	GPIO_PP0 - GPIO_PP7
GPIO5	Port Q	0x200	GPIO_PQ0 - GPIO_PQ7
	Port R	0x204	GPIO_PR0 - GPIO_PR7
	Port S	0x208	GPIO_PS0 - GPIO_PS7
	Port T	0x20C	GPIO_PT0 - GPIO_PT7
GPIO6	Port U	0x280	GPIO_PU0 - GPIO_PU7
	Port V	0x284	GPIO_PV0 - GPIO_PV7
	Port W	0x288	GPIO_PW0 - GPIO_PW7
	Port X	0x28C	GPIO_PX0 - GPIO_PX7
GPIO7	Port Y	0x300	GPIO_PY0 - GPIO_PY7
	Port Z	0x304	GPIO_PZ0 - GPIO_PZ7
	Port AA	0x308	GPIO_PAA0 - GPIO_PAA7
	Port BB	0x30C	GPIO_PBB0 - GPIO_PBB7

Use the appropriate GPIO_CNF register to designate a pin to act either as a GPIO as an SFIO. Setting a pin's CNF bit to 1 configures the pin to be used as a GPIO and prevents the pin from being used as for its Primary, Alternate 1, Alternate 2, or Alternate 3 SFIO function. Setting the CNF bit to 0 allows the port to be used as an SFIO.

For each pin designated to act as a GPIO, the corresponding bit in the appropriate GPIO_OE register controls the pin's output enable. When the OE bit is 1, the pin is driven LOW or HIGH according to the state of the corresponding OUT bit in the appropriate GPIO_OUT register. When the OE bit is 0, the output pin is tri-stated and can be used as an input.

Each IN bit in each GPIO_IN register contains the registered input value of a pin designated as a GPIO. The input goes through two D flip-flops to protect against meta-stability. The following figure shows how the pins are shared and multiplexed functionally.

Figure 32 SFIO/GPIO Pin Multiplexing Architecture



The SFIO output enable signals are generated automatically by the modules controlling the signals; they are not controlled through registers. However, the SFIO output enable mux does select which group of SFIO output enables signals to use based on the bits in the PIN_MUX_CTL registers.

These SFIO output enables are then sent to another mux that is controlled with the bits in the GPIO_CNF registers. Next, the selected output enables are ANDed with the bits in the TRI_STATE register. The ANDing function allows the output pins to be driven only if the TRI_STATE register bit is disabled.

In the input direction, the state of the input buffers is controlled with the DATAIN_EN signals. These signals are generated automatically by the modules controlling the signals; they are not controlled through registers. The first mux selects between the SFIO and GPIO input signals, and the last mux passes on the appropriate group of SFIO signals if so allowed.

18.2 GPIO Registers

Each port of each GPIO controller has several registers for the control and monitoring of the port's 8 GPIO pins. The registers provide per-pin control of the following features:

- GPIO_CNF_* / GPIO_MSK_CNF — Select SPIO or GPIO
- GPIO_OE_* / GPIO_MSK_OE_* — Output Enable
- GPIO_OUT_* / GPIO_MSK_OUT_* — GPIO Output Value
- GPIO_IN_* — GPIO Input Value (Read Only)
- GPIO_INT_STA_* / GPIO_MSK_INT_STA_* — GPIO Interrupt Status
- GPIO_INT_ENB_* / GPIO_MSK_INT_ENB_* — Interrupt Enable
- GPIO_INT_LVL_* / GPIO_MSK_INT_LVL_* — Interrupt Selection (Edge/Level)
- GPIO_INT_CLR_* — Interrupt Flag Set-to-Clear

Many of the control registers are accessible from two offsets. Accesses to the lower offsets affect all 8 pins for the GPIO port. Accesses to the upper offsets provide a per-pin mask capability allowing adjustments to a subset of the pins. Using the upper offsets can eliminate the need for Read-Modify-Write operations. For example, a write to GPIO_MSK_CNF can substitute for a Read-Modify-Write of GPIO_CNF.

Table 61 Tegra GPIO Register Summary (Base Address: 6000.Dxxxh)

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	800	804	808	80C
GPIO_OE_0	010	014	018	01C	810	814	818	81C
GPIO_OUT_0	020	024	028	02C	820	824	828	82C
GPIO_IN_0	030	034	038	03C				
GPIO_INT_STA_0	040	044	048	04C	840	844	848	84C
GPIO_INT_ENB_0	050	054	058	05C	850	854	858	85C
GPIO_INT_LVL_0	060	064	068	06C	860	864	868	86C
GPIO_INT_CLR_0	070	074	078	07C				
GPIO Controller 2 – Port	**E**	**F**	**G**	**H**	**E**	**F**	**G**	**H**
GPIO_CNF_1	080	084	088	08C	880	884	888	88C
GPIO_OE_1	090	094	098	09C	890	894	898	89C
GPIO_OUT_1	0A0	0A4	0A8	0AC	8A0	8A4	8A8	8AC
GPIO_IN_1	0B0	0B4	0B8	0BC				
GPIO_INT_STA_1	0C0	0C4	0C8	0CC	8C0	8C4	8C8	8CC
GPIO_INT_ENB_1	0D0	0D4	0D8	0DC	8D0	8D4	8D8	8DC
GPIO_INT_LVL_1	0E0	0E4	0E8	0EC	8E0	8E4	8E8	8EC
GPIO_INT_CLR_1	0F0	0F4	0F8	0FC				
GPIO Controller 3 – Port	**I**	**J**	**K**	**L**	**I**	**J**	**K**	**L**
GPIO_CNF_2	100	104	108	10C	900	904	908	90C
GPIO_OE_2	110	114	118	11C	910	914	918	91C
GPIO_OUT_2	120	124	128	12C	920	924	928	92C
GPIO_IN_2	130	134	138	13C				

Register Name	Offset (Lower: Read-Modify-Write)				Offset (Upper: Per-Pin Mask Write)			
GPIO_INT_STA_2	140	144	148	14C	940	944	948	94C
GPIO_INT_ENB_2	150	154	158	15C	950	954	958	95C
GPIO_INT_LVL_2	160	164	168	16C	960	964	968	96C
GPIO_INT_CLR_2	170	174	178	17C				
GPIO Controller 4 – Port	**M**	**N**	**O**	**P**	**M**	**N**	**O**	**P**
GPIO_CNF_3	180	184	188	18C	980	984	988	98C
GPIO_OE_3	190	194	198	19C	990	994	998	99C
GPIO_OUT_3	1A0	1A4	1A8	1AC	9A0	9A4	9A8	9AC
GPIO_IN_3	1B0	1B4	1B8	1BC				
GPIO_INT_STA_3	1C0	1C4	1C8	1CC	9C0	9C4	9C8	9CC
GPIO_INT_ENB_3	1D0	1D4	1D8	1DC	9D0	9D4	9D8	9DC
GPIO_INT_LVL_3	1E0	1E4	1E8	1EC	9E0	9E4	9E8	9EC
GPIO_INT_CLR_3	1F0	1F4	1F8	1FC				
GPIO Controller 5 – Port	**Q**	**R**	**S**	**T**	**Q**	**R**	**S**	**T**
GPIO_CNF_4	200	204	208	20C	A00	A04	A08	A0C
GPIO_OE_4	210	214	218	21C	A10	A14	A18	A1C
GPIO_OUT_4	220	224	228	22C	A20	A24	A28	A2C
GPIO_IN_4	230	234	238	23C				
GPIO_INT_STA_4	240	244	248	24C	A40	A44	A48	A4C
GPIO_INT_ENB_4	250	254	258	25C	A50	A54	A58	A5C
GPIO_INT_LVL_4	260	264	268	26C	A60	A64	A68	A6C
GPIO_INT_CLR_4	270	274	278	27C				
GPIO Controller 6 – Port	**U**	**V**	**W**	**X**	**U**	**V**	**W**	**X**
GPIO_CNF_5	280	284	288	28C	A80	A84	A88	A8C
GPIO_OE_5	290	294	298	29C	A90	A94	A98	A9C
GPIO_OUT_5	2A0	2A4	2A8	2AC	AA0	AA4	AA8	AAC
GPIO_IN_5	2B0	2B4	2B8	2BC				
GPIO_INT_STA_5	2C0	2C4	2C8	2CC	AC0	AC4	AC8	ACC
GPIO_INT_ENB_5	2D0	2D4	2D8	2DC	AD0	AD4	AD8	ADC
GPIO_INT_LVL_5	2E0	2E4	2E8	2EC	AE0	AE4	AE8	AEC
GPIO_INT_CLR_5	2F0	2F4	2F8	2FC				
GPIO Controller 7 – Port	**Y**	**Z**	**AA**	**BB**	**Y**	**Z**	**AA**	**BB**
GPIO_CNF_6	300	304	308	30C	B00	B04	B08	B0C
GPIO_OE_6	310	314	318	31C	B10	B14	B18	B1C
GPIO_OUT_6	320	324	328	32C	B20	B24	B28	B2C
GPIO_IN_6	330	334	338	33C				
GPIO_INT_STA_6	340	344	348	34C	B40	B44	B48	B4C
GPIO_INT_ENB_6	350	354	358	35C	B50	B54	B58	B5C
GPIO_INT_LVL_6	360	364	368	36C	B60	B64	B68	B6C
GPIO_INT_CLR_6	370	374	378	37C				

18.2.1 GPIO_CNF_0

Designate whether each pin operates as a GPIO or as an SFIO. By default all Pins come up in SFIO mode. These can be programmed to GPIO mode at any stage.

GPIO Port Configuration Registers

Offset: 000h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	BIT_7: Configures each pin to be in either GPIO or SFIO mode 0 = SPIO 1 = GPIO
6	0x0	BIT_6: Configures each pin to be in either GPIO or SFIO mode 0 = SPIO 1 = GPIO
5	0x0	BIT_5: Configures each pin to be in either GPIO or SFIO mode 0 = SPIO 1 = GPIO
4	0x0	BIT_4: Configures each pin to be in either GPIO or SFIO mode 0 = SPIO 1 = GPIO
3	0x0	BIT_3: Configures each pin to be in either GPIO or SFIO mode 0 = SPIO 1 = GPIO
2	0x0	BIT_2: Configures each pin to be in either GPIO or SFIO mode 0 = SPIO 1 = GPIO
1	0x0	BIT_1: Configures each pin to be in either GPIO or SFIO mode 0 = SPIO 1 = GPIO
0	0x0	BIT_0: Configures each pin to be in either GPIO or SFIO mode 0 = SPIO 1 = GPIO

18.2.2 GPIO_OE_0

Selectively enable the output drivers for pins designated as GPIOs (i.e. GPIO_CNF.x=1).

GPIO Port Enable Registers

Offset: 010h | 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

Bit	Reset	Description
5	0x0	BIT_5: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = TRI_STATE 1 = DRIVEN
4	0x0	BIT_4: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = TRI_STATE 1 = DRIVEN
3	0x0	BIT_3: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = TRI_STATE 1 = DRIVEN
2	0x0	BIT_2: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = TRI_STATE 1 = DRIVEN
1	0x0	BIT_1: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = TRI_STATE 1 = DRIVEN
0	0x0	BIT_0: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = TRI_STATE 1 = DRIVEN

18.2.3 GPIO_OUT_0

Specify the value driven from each pin that is designated as a GPIO (i.e. GPIO_CNF.x=1) and whose output is enabled (i.e. GPIO_OE.x=1).

GPIO Port Output Value Registers (Read/Write)

Offset: 020h | 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) mxst 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) mxst 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) mxst 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) mxst 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) mxst be true for this to be a valid state 0 = LOW 1 = HIGH

Bit	Reset	Description
2	0x0	BIT_2: GPIO_CNF.x=1 (in GPIO mode) AND GPIO_OE.x=1 (GPIO output enabled) mxst 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) mxst 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) mxst be true for this to be a valid state 0 = LOW 1 = HIGH

18.2.4 GPIO_IN_0 GPIO

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.

GPIO Port Input Value Registers (Read Only)

Offset: 030h | 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

Each GPIO input pin can be programmed to generate an interrupt when the state of the pin

- Changes from low to high
- Changes from high to low
- Is low
- Is high
- Changes from high to low OR low to high

Each GPIO controller forwards a single interrupt signal to the system interrupt controller. The controller-level interrupt is the logical OR of the interrupts from its 4 ports. Similarly, the interrupt from a port is the logical OR of the interrupts from its 8 pins. The system interrupt controller provides a mechanism for masking each GPIO controller's interrupt.

18.2.5 GPIO_INT_STA_0

GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. Every GPIO pin generates and Interrupt when switching from Low-High ot High-Low. Interrupt status for each port is saved in an Interrupt status registers

GPIO Port Interrupt Status Registers

Offset: 040h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	BIT_7: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = IN_ACTIVE = ACTIVE
6	0x0	BIT_6: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = IN_ACTIVE 1 = ACTIVE
5	0x0	BIT_5: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = IN_ACTIVE 1 = ACTIVE
4	0x0	BIT_4: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = IN_ACTIVE 1 = ACTIVE
3	0x0	BIT_3: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = IN_ACTIVE 1 = ACTIVE
2	0x0	BIT_2: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = IN_ACTIVE 1 = ACTIVE
1	0x0	BIT_1: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = IN_ACTIVE 1 = ACTIVE

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 = IN_ACTIVE 1 = ACTIVE

18.2.6 GPIO_INT_ENB_0

Selectively enable the interrupt functionality for each pin designated as a GPIO (GPIO_CNF.x=1). The event/state which triggers a pin's interrupt is controlled via the GPIO_INT_LVL register.

GPIO Port Interrupt Enable Registers

Offset: 050h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	BIT_7: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. 0 = DISABLE 1 = ENABLE
6	0x0	BIT_6: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. 0 = DISABLE 1 = ENABLE
5	0x0	BIT_5: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. 0 = DISABLE 1 = ENABLE
4	0x0	BIT_4: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. 0 = DISABLE 1 = ENABLE
3	0x0	BIT_3: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. 0 = DISABLE 1 = ENABLE
2	0x0	BIT_2: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. 0 = DISABLE 1 = ENABLE
1	0x0	BIT_1: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. 0 = DISABLE 1 = ENABLE
0	0x0	BIT_0: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. 0 = DISABLE 1 = ENABLE

18.2.7 GPIO_INT_LVL_0

For each pin enabled as a GPIO and enabled for a GPIO interrupt, specify the pin state or pin state change which triggers the interrupt. Bits [8:15] select between edge and level triggered interrupt on a per-pin basis.

For level-triggered interrupts, bits [0:7] choose whether a logic high or a logic low triggers an interrupt.

For edge-triggered interrupts, bits [16:23] allow for interrupt on any change in pin state.

For edge-triggered interrupts whose DELTA bit is clear, bits [0:7] choose whether a rising edge or a falling edge on the pin triggers an interrupt.

GPIO Port Interrupt Activation Level Registers

Offset: 060h | 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
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

Bit	Reset	Description
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

18.2.8 GPIO_INT_CLR_0

This register clears the Interrupts which are set. This is a Write-Only register. This is valid only in GPIO mode and GPIO_INT.ENB is set.

GPIO Port Interrupt Flag Set-to-Clear Registers

Offset: 070h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	BIT_7: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = SET 1 = CLEAR
6	0x0	BIT_6: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = SET 1 = CLEAR
5	0x0	BIT_5: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = SET 1 = CLEAR
4	0x0	BIT_4: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = SET 1 = CLEAR
3	0x0	BIT_3: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = SET 1 = CLEAR
2	0x0	BIT_2: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = SET 1 = CLEAR
1	0x0	BIT_1: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = SET 1 = CLEAR
0	0x0	BIT_0: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = SET 1 = CLEAR

18.2.9 GPIO_MSK_CNF_0

This write-only pseudo-register allows for updates of individual bits in the GPIO_CNF register.

MASKED PRIMARY GPIO/SFIO Config Registers (Masked Writes)

Offset: 800h | Read/Write: R/W | Reset: 0b0000000000000000

Bit	R/W	Reset	Description
15	WO	0x0	MSK7: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
14	WO	0x0	MSK6: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
13	WO	0x0	MSK5: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
12	WO	0x0	MSK4: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
11	WO	0x0	MSK3: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
10	WO	0x0	MSK2: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
9	WO	0x0	MSK1: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
8	WO	0x0	MSK0: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
7	RW	0x0	BIT_7: 0 = SPIO 1 = GPIO
6	RW	0x0	BIT_6: 0 = SPIO 1= GPIO
5	RW	0x0	BIT_5: 0 = SPIO 1 = GPIO
4	RW	0x0	BIT_4: 0 = SPIO 1 = GPIO
3	RW	0x0	BIT_3: 0 = SPIO 1 = GPIO

Bit	R/W	Reset	Description
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

18.2.10 GPIO_MSK_OE_0

This write-only pseudo-register allows for updates of individual bits in the GPIO_OE register.

GPIO Masked Output Enable (Masked Writes)

Offset: 810h | 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

Bit	R/W	Reset	Description
6	RW	0x0	BIT_6: 0 = TRI_STATE 1 = DRIVEN
5	RW	0x0	BIT_5: 0 = TRI_STATE 1 = DRIVEN
4	RW	0x0	BIT_4: 0 = TRI_STATE 1 = DRIVEN
3	RW	0x0	BIT_3: 0 = TRI_STATE 1 = DRIVEN
2	RW	0x0	BIT_2: 0 = TRI_STATE 1 = DRIVEN
1	RW	0x0	BIT_1: 0 = TRI_STATE 1 = DRIVEN
0	RW	0x0	BIT_0: 0 = TRI_STATE 1 = DRIVEN

18.2.11 GPIO_MSK_OUT_0

This write-only pseudo-register allows for updates of individual bits in the GPIO_OUT register.

GPIO A-D Masked Output Enable (Masked Writes)

Offset: 820h | Read/Write: R/W | Reset: 0b0000000000000000

Bit	R/W	Reset	Description
15	WO	0x0	MSK7: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
14	WO	0x0	MSK6: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
13	WO	0x0	MSK5: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
12	WO	0x0	MSK4: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
11	WO	0x0	MSK3: 0=Disable bit for write 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
10	WO	0x0	MSK2: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
9	WO	0x0	MSK1: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
8	WO	0x0	MSK0: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
7	RW	0x0	BIT_7: 0 = LOW 1 = HIGH
6	RW	0x0	BIT_6: 0 = LOW 1 = HIGH
5	RW	0x0	BIT_5: 0 = LOW 1 = HIGH
4	RW	0x0	BIT_4: 0 = LOW 1 = HIGH
3	RW	0x0	BIT_3: 0 = LOW 1 = HIGH
2	RW	0x0	BIT_2: 0 = LOW 1 = HIGH
1	RW	0x0	BIT_1: 0 = LOW 1 = HIGH
0	RW	0x0	BIT_0: 0 = LOW 1 = HIGH

18.2.12 GPIO_MSK_INT_STA_0

This write-only pseudo-register is functionally equivalent to GPIO_INT_CLR.

GPIO A-D Masked Interrupt Status (Masked Clears)

Offset: 840h | Read/Write: R/W | Reset: 0b0000000000000000

Bit	R/W	Reset	Description
15	WO	0x0	MSK7: 0=Disable bit for write 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
14	WO	0x0	MSK6: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
13	WO	0x0	MSK5: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
12	WO	0x0	MSK4: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
11	WO	0x0	MSK3: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
10	WO	0x0	MSK2: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
9	WO	0x0	MSK1: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
8	WO	0x0	MSK0: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
7	RW	0x0	BIT_7: 0 = IN_ACTIVE 1 = ACTIVE
6	RW	0x0	BIT_6: 0 = IN_ACTIVE 1 = ACTIVE
5	RW	0x0	BIT_5: 0 = IN_ACTIVE 1 = ACTIVE
4	RW	0x0	BIT_4: 0 = IN_ACTIVE 1 = ACTIVE
3	RW	0x0	BIT_3: 0 = IN_ACTIVE 1 = ACTIVE
2	RW	0x0	BIT_2: 0 = IN_ACTIVE 1 = ACTIVE
1	RW	0x0	BIT_1: 0 = IN_ACTIVE 1 = ACTIVE
0	RW	0x0	BIT_0: 0 = IN_ACTIVE 1 = ACTIVE

18.2.13 GPIO_MSK_INT_ENB_0

This write-only pseudo-register allows for updates of individual bits in the GPIO_INT_ENB register.

GPIO A-D Masked Interrupt Enable (Masked Writes)

Offset: 850h | Read/Write: R/W | Reset: 0b0000000000000000

Bit	R/W	Reset	Description
15	WO	0x0	MSK7: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
14	WO	0x0	MSK6: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
13	WO	0x0	MSK5: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
12	WO	0x0	MSK4: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
11	WO	0x0	MSK3: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
10	WO	0x0	MSK2: 0=Disable bit for write 1 = ENABLE
9	WO	0x0	MSK1: 0=Disable bit for write 1 = ENABLE
8	WO	0x0	MSK0: 0=Disable bit for write 1 = ENABLE
7	RW	0x0	BIT_7: 0 = DISABLE 1 = ENABLE
6	RW	0x0	BIT_6: 0 = DISABLE 1 = ENABLE
5	RW	0x0	BIT_5: 0 = DISABLE 1 = ENABLE
4	RW	0x0	BIT_4: 0 = DISABLE 1 = ENABLE
3	RW	0x0	BIT_3: 0 = DISABLE 1 = ENABLE
2	RW	0x0	BIT_2: 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
1	RW	0x0	BIT_1: 0 = DISABLE 1 = ENABLE
0	RW	0x0	BIT_0: 0 = DISABLE 1 = ENABLE

18.2.14 GPIO_MSK_INT_LVL_0

This write-only pseudo-register allows for updates of individual bits in the GPIO_INT_LVL register.

GPIO Masked Write Interrupt Activation Levels

Offset: 860h | Read/Write: R/W | Reset: 0b0000000000000000

Bit	R/W	Reset	Description
15	WO	0x0	MSK7: 0=Disable bit for write 1 = ENABLE
14	WO	0x0	MSK6: 0=Disable bit for write 1 = ENABLE
13	WO	0x0	MSK5: 0=Disable bit for write 1 = ENABLE
12	WO	0x0	MSK4: 0=Disable bit for write 1 = ENABLE
11	WO	0x0	MSK3: 0=Disable bit for write 1 = ENABLE
10	WO	0x0	MSK2: 0=Disable bit for write 1 = ENABLE
9	WO	0x0	MSK1: 0=Disable bit for write 1 = ENABLE
8	WO	0x0	MSK0: 0=Disable bit for write 1 = ENABLE
7	RW	0x0	BIT_7: 0 = LOW 1 = HIGH
6	RW	0x0	BIT_6: 0 = LOW 1 = HIGH
5	RW	0x0	BIT_5: 0 = LOW 1 = HIGH
4	RW	0x0	BIT_4: 0 = LOW 1 = HIGH

Bit	R/W	Reset	Description
3	RW	0x0	BIT_3: 0 = LOW 1 = HIGH
2	RW	0x0	BIT_2: 0 = LOW 1 = HIGH
1	RW	0x0	BIT_1: 0 = LOW 1 = HIGH
0	RW	0x0	BIT_0: 0 = LOW 1 = HIGH

19.0 KEYBOARD CONTROLLER

The Keyboard Controller (KBC) is used to interface a maximum of 24-pins with an external keypad. The number of pins used in the KBC depends on the external keypad row and column count, and is less in some Tegra® 2 Series package variations.

A keyboard controller for a keypad matrix implemented completely in software can result in significant overhead. The KBC lowers the burden on software by supporting keypad scan, de-bounce and wake-up on any key-press in hardware. It also reduces power consumption in keypad associated operations. The remaining pins can be used as GPIO. If no external keypad is connected all the pins can be used as GPIO. The KBC module is a slave on APB bus.

The typical software scanning flow is shown in Figure 33. Most of the tasks in the flowchart can be done in hardware, off-loading them from software. The three main operations done by KBC are:

- Keypad scan
- De-bounce
- Control SM

In the presence of a KBC, the software flowchart for keyboard control is reduced to what is shown in Figure 34. The software only needs to configure the keyboard controller and serve the interrupts. This automatically reduces the power consumption related to keypad operations.

Features

The KBC supports:

- Keypad matrix size of up to 16X8.
- Wake-up on interrupt mode
- Continuous polling mode
- Multi-key press support (dependency on keypad HW)
- Joystick (16ms key-press speed)

The keypad matrix provides ghost key protection in case of three or more simultaneous key presses. The system has a power ON/OFF key. The handling of this key is a little different from the rest of the keys. KBC is not aware of this difference and treats it as any other key; different modules in the Tegra 2 Series devices (e.g., Always ON) monitor this key when the device is powered off.

Row 0 should always be enabled. Rows that are enabled must be contiguous. For example, if we need to enable three rows, then they must be rows 0, 1 and 2. Enabling rows 0, 1, and 3, for example, is an invalid combination since the rows are not contiguous.

Figure 33 Typical Software Scanning Flow

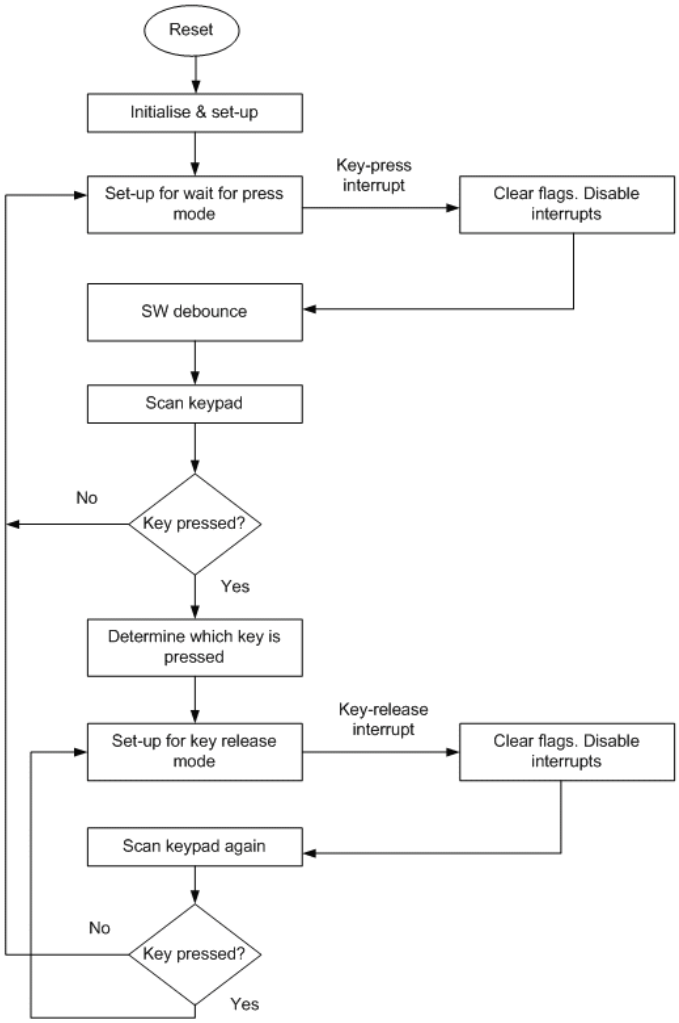


Figure 34 Software Flow with Hardware Keyboard Controller

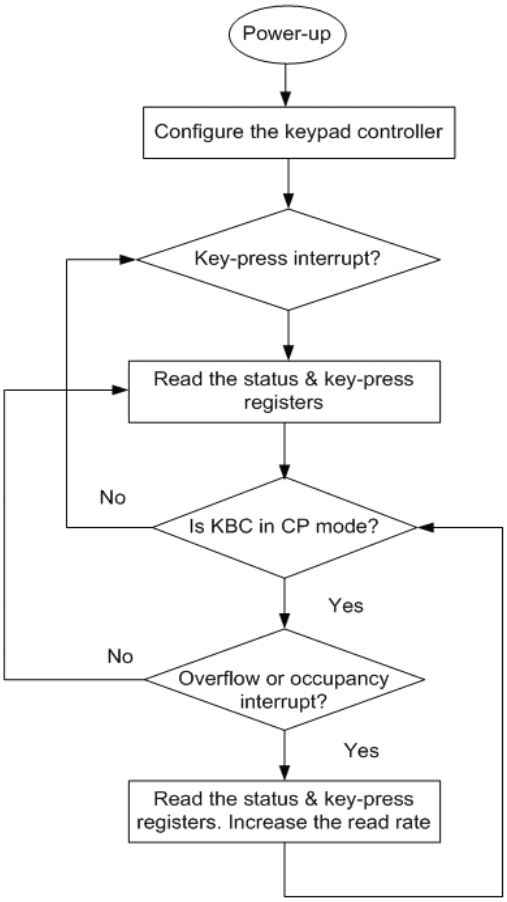
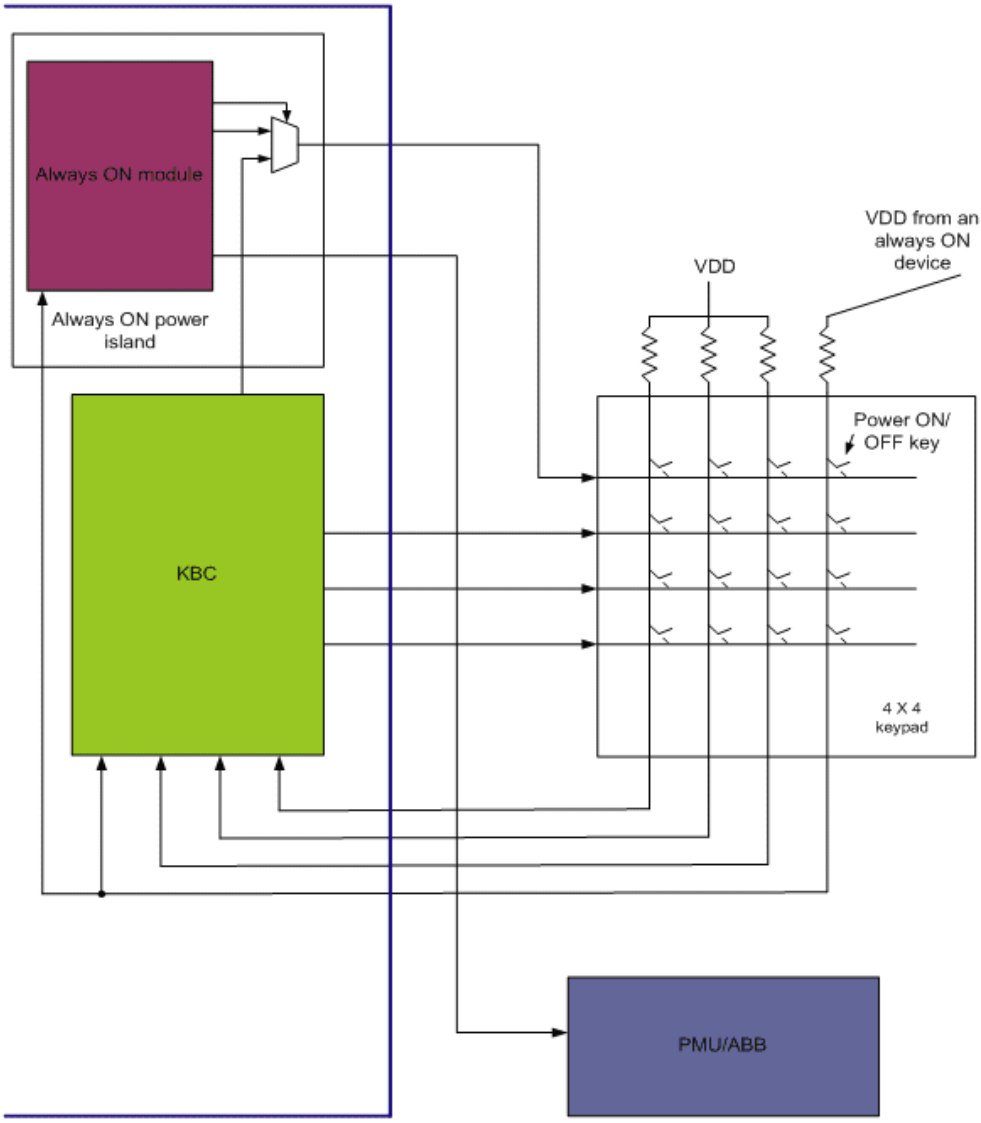


Figure 35 Power ON/OFF Key Example

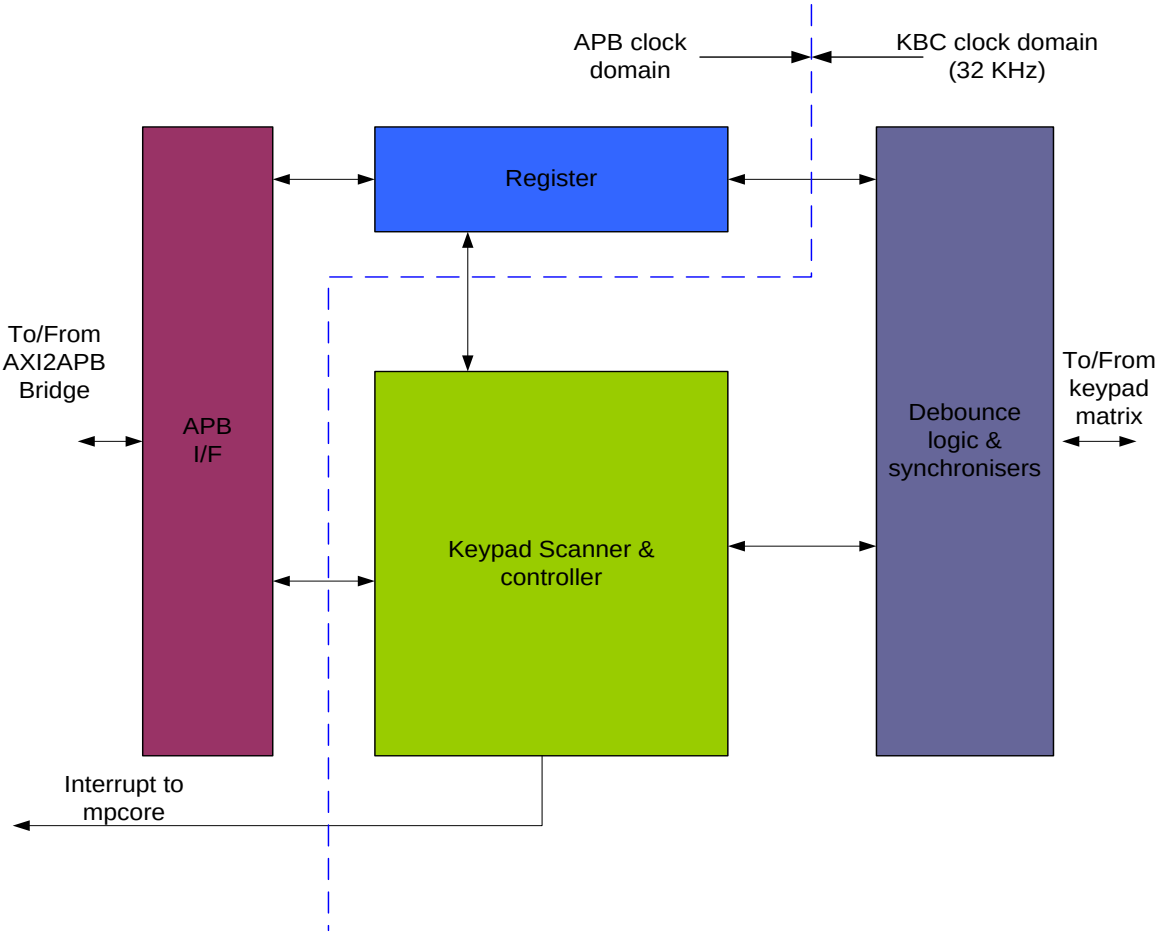


Clocking

- APB clock for APB interface and register sub-module
- KBC clock for keypad scanner and controller (32 KHz)

19.1 Functionality

Figure 36. KBC Functional Block Diagram



Keyboard controller is enabled when the enable bit is set and some pins are configured for interfacing with keypad matrix. The pins which are not interfaced to keypad matrix can be used as GPIO pins. To configure KBC, the following registers should be programmed:

- APBDEV_KBC_ROW_CFG
- APBDEV_KBC_INT
- APBDEV_KBC_COL_CFG
- APBDEV_KBC_CONTROL

All the pins interfacing with keypad use de-bounce logic to detect a valid key press (filter out glitches).

Initially, the KBC is in wake-up state on key-press (WuKP). In this mode, it waits for any key-press. All rows are driven low to detect any key-press in this state. Once a key is pressed, KBC generates an optional key-press interrupt (KP_INT_STATUS in APBDEV_KBC_INT register), goes to continuous polling (CP) state and starts scanning after INIT_DLY_VAL (in APBDEV_KBC_INIT_DLY register) cycles.

In CP state, the delay between two consecutive full scans is RPT_DLY_VAL (in APBDEV_KBC_RPT_DLY register) cycles. In CP state, it scans all the rows one-by-one by driving one row low while other rows are configured as input so they are driven "Z". Then all the columns are sampled for each row scan.

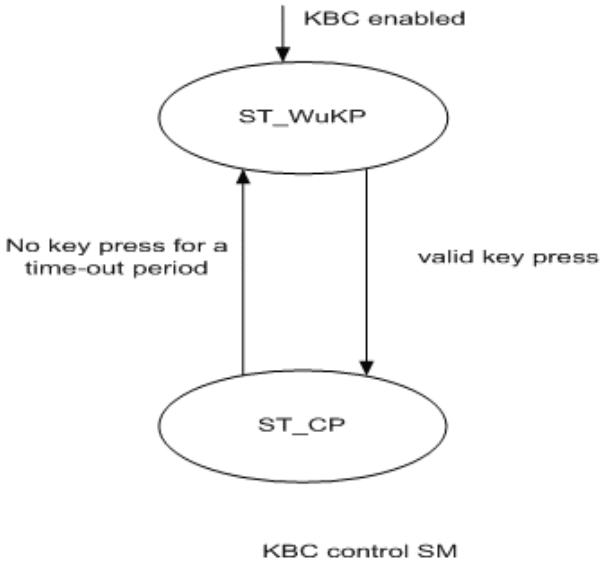
The driving of rows and sampling of columns is based on de-bounce cycle. The KBC cycle is 32 KHz (time period = 31.25 microseconds). So if de-bounce count is 4, one row scan takes 125 microseconds. One complete scan of all rows (16 max.) takes 2 milliseconds. In one second there can be 500 complete scans (if RPT_DLY_VAL = 0).

KBC supports up to eight simultaneous key-presses per complete scan. These eight key numbers are stored in the registers APBDEV_KBC_KP_ENT0 and APBDEV_KBC_KP_ENT1. The FIFO pointer increments on the second register entry register (APBDEV_KBC_KP_ENT1) read. Software should always read the two entry registers consecutively. It should first read APBDEV_KBC_KP_ENT0 and then read APBDEV_KBC_KP_ENT1.

To avoid any wrong key-press more than once, software should see the AV_FIFO_CNT value in APBDEV_KBC_INT register. The entry registers give new key press information only when AV_FIFO_CNT value is non-zero. A zero value indicates that there are no new key presses after the last read. To indicate whether an entry is valid or not, a valid bit is associated with every entry in the two entry registers.

Software should read the entry register fast enough in CP state to avoid overflow. To avoid an overflow in case of slow reads, the first level warning indicator can be programmed in FIFO_TH_CNT field of the APBDEV_KBC_CONTROL register. An optional interrupt (FIFO_CNT_INT_STATUS in APBDEV_KBC_INT register) can be generated when the FIFO occupancy count reaches this value. In case of overflow an optional interrupt (FIFO_OVF_INT_STATUS in APBDEV_KBC_INT register) is generated and software will have two options of treating the new entries. It can either drop them or overwrite them. This bit is FIFO_MODE in APBDEV_KBC_CONTROL register. Figure below shows KBC control the SM.

Figure 37 KBC Control Logic



KBC control SM

19.2 Micro-Architecture

19.2.1 Keypad Matrix

The maximum supported keypad matrix size varies by the Tegra 2 Series package variation.

The KBC controller itself can support up to a 16x8 keypad matrix (16 rows and 8 columns). Up to eight simultaneous key presses can be stored per full scan (row scanning). Storage for key press events is as follows: an asynchronous FIFO stores 1 key press event, a synchronous FIFO stores eight events and a register stores 1 event.

A minimum of two rows must always be enabled.

19.2.2 Additional Interface Signals

kbc_interrrupt: An interrupt in the 32 KHz domain is sent to PMC and is used to wake up the system.

19.2.3 Pull-ups and pull-downs in pads

All the pads have pull-ups and pull-down bits which can be programmed by the software per your requirement(s). Contact your NVIDIA representative for details on the pullup/pulldown registers.

Rows are always output to the Tegra 2 Series and Columns are inputs. Row and Columns for the IO can be programmed by software as needed.

19.2.4 Ghost Key Detection

Ghost key detection is built-in to the way scanning interacts with keys that are simple switches. The solution is to have a diode in series with the key switch to avoid the ghost paths.

19.3 KBC Registers

19.3.1 APBDEV_KBC_CONTROL_0

This register is the main control register. It should be configured last after configuring all other settings.

KBC Control Register

Offset: 000h | Read/Write: R/W | Reset: 0b0010000000000000000

Bit	Reset	Description
18	0x0	FIFO_MODE: Selects the behavior in case of FIFO overflow 0 = Drop the new detected key-presses 1 = Overwrite the new detected key-presses
17:14	0x4	FIFO_TH_CNT: FIFO threshold count. Keeps the threshold FIFO occupancy count. If FIFO reaches/crosses that count an optional interrupt will be raised. Should not be programmed as 0 N = Threshold occupancy count is N
13:4	0x0	DBC_CNT: De-bounce count. This value sets the De-bounce FSM associated with each KBC input pin evaluate the input transitions 0 = No De-bounce N = N KBC clocks
3	0x0	FIFO_CNT_INT_EN: FIFO threshold count interrupt enable. Setting this bit will enable interrupt when FIFO occupancy reaches/crosses the value specified in FIFO_TH_CNT 0 Disable FIFO overflow interrupt 1 Enable FIFO overflow interrupt
2	0x0	FIFO_OVF_INT_EN: FIFO overflow interrupt enable. Setting this bit will enable interrupt on FIFO overflow
1	0x0	KP_INT_EN: Key-press interrupt enable. Setting this bit will enable interrupt on any key-press 0 = DISABLE 1 = ENABLE
0	0x0	EN: Keyboard controller enable. Setting this bit will override the pins settings done in GPIO 0 = DISABLE 1 = ENABLE

19.3.2 APBDEV_KBC_INT_0

KBC Interrupt Status and Clear Register

Offset: 004h | Read/Write: R/W | Reset: 0bxxxxx000

Bit	R/W	Reset	Description
7:4	RO	X	AV_FIFO_CNT: FIFO occupancy count. Shows the number of unread registers. Read only.
3	RO	X	KBC_ST_STATUS: KBC status. Read only. 0 = WuKP (Wake-up on key-press) interrupt mode 1 = CP (Continuous polling) mode
2	RW	0x0	FIFO_CNT_INT_STATUS: FIFO threshold count interrupt status. Writing '1' to this bit will clear the interrupt 0 FIFO threshold count interrupt de-asserted 1 FIFO threshold count interrupt asserted (read) 1 Clear FIFO overflow interrupt (write)
1	RW	0x0	FIFO_OVF_INT_STATUS: FIFO overflow interrupt status. Writing '1' to this bit will clear the interrupt 0 FIFO overflow interrupt de-asserted 1 FIFO overflow interrupt asserted (read) 1 Clear FIFO overflow interrupt (write)
0	RW	0x0	KP_INT_STATUS: Key-press interrupt status. Writing '1' to this bit will clear the interrupt 0 Key-press interrupt de-asserted 1 Key-press interrupt asserted (read) 1 Clear key-press interrupt (write)

19.3.3 APBDEV_KBC_ROW_CFG0_0

This register setting will take precedence in case same GPIO pin is configured in column register also. It does the mapping between GPIO pins and rows of keypad matrix. It configures for GPIO pin# 0 to 5.

First Row Configuration Register

Offset: 008h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
29:26	0x0	GPIO_5_ROW_NUM: Mapping of GPIO pin# 5 to row number. Valid only if GPIO_5_ROW_EN is set. Indicates row number 0x0 = Row number 0 0xF = Row number 15
25	0x0	GPIO_5_ROW_EN: Indicates whether GPIO pin# 5 is mapped to any row of keypad matrix. This bit overrides any setting done for pin# 5 in column configuration 0 = NOT_MAPPED 1 = MAPPED
24:21	0x0	GPIO_4_ROW_NUM: Mapping of GPIO pin# 4 to row number. Valid only if GPIO_4_ROW_EN is set. Indicates row number 0x0 = Row number 0 0xF = Row number 15
20	0x0	GPIO_4_ROW_EN: Indicates whether GPIO pin# 4 is mapped to any row of keypad matrix. This bit overrides any setting done for pin# 4 in column configuration 0 = NOT_MAPPED 1 = MAPPED
19:16	0x0	GPIO_3_ROW_NUM: Mapping of GPIO pin# 3 to row number. Valid only if GPIO_3_ROW_EN is set. Indicates row number 0x0 = Row number 0 0xF = Row number 15
15	0x0	GPIO_3_ROW_EN: Indicates whether GPIO pin# 3 is mapped to any row of keypad matrix. This bit overrides any setting done for pin# 3 in column configuration 0 = NOT_MAPPED 1 = MAPPED

Bit	Reset	Description
14:11	0x0	GPIO_2_ROW_NUM: Mapping of GPIO pin# 2 to row number. Valid only if GPIO_2_ROW_EN is set. Indicates row number 0x0 = Row number 0 0xF = Row number 15
10	0x0	GPIO_2_ROW_EN: Indicates whether GPIO pin# 2 is mapped to any row of keypad matrix. This bit overrides any setting done for pin# 2 in column configuration 0 = NOT_MAPPED 1 = MAPPED
9:6	0x0	GPIO_1_ROW_NUM: Mapping of GPIO pin# 1 to row number. Valid only if GPIO_1_ROW_EN is set. Indicates row number 0x0 = Row number 0 0xF = Row number 15
5	0x0	GPIO_1_ROW_EN: Indicates whether GPIO pin# 1 is mapped to any row of keypad matrix. This bit overrides any setting done for pin# 1 in column configuration 0 = NOT_MAPPED 1 = MAPPED
4:1	0x0	GPIO_0_ROW_NUM: Mapping of GPIO pin# 0 to row number. Valid only if GPIO_0_ROW_EN is set. Indicates row number 0x0 = Row number 0 0xF = Row number 15
0	0x0	GPIO_0_ROW_EN: Indicates whether GPIO pin# 0 is mapped to any row of keypad matrix. This bit overrides any setting done for pin# 0 in column configuration 0 = NOT_MAPPED 1 = MAPPED

19.3.4 APBDEV_KBC_ROW_CFG1_0

This register setting takes precedence in case same GPIO pin is configured in column register also. It does the mapping between GPIO pins and rows of keypad matrix. It configures for GPIO pin# 6 to 11.

Second Row Configuration Register

Offset: 00ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
29:26	0x0	GPIO_11_ROW_NUM: Mapping of GPIO pin# 11 to row number. Valid only if GPIO_11_ROW_EN is set. Indicates row number 0x0 = Row number 0 0xF = Row number 15
25	0x0	GPIO_11_ROW_EN: Indicates whether GPIO pin# 11 is mapped to any row of keypad matrix. This bit overrides any setting done for pin# 11 in column configuration 0 = NOT_MAPPED 1 = MAPPED
24:21	0x0	GPIO_10_ROW_NUM: Mapping of GPIO pin# 10 to row number. Valid only if GPIO_10_ROW_EN is set. Indicates row number 0x0 = Row number 0 0xF = Row number 15
20	0x0	GPIO_10_ROW_EN: Indicates whether GPIO pin# 10 is mapped to any row of keypad matrix. This bit overrides any setting done for pin# 10 in column configuration 0 = NOT_MAPPED 1 = MAPPED
19:16	0x0	GPIO_9_ROW_NUM: Mapping of GPIO pin# 9 to row number. Valid only if GPIO_9_ROW_EN is set. Indicates row number 0x0 = Row number 0 0xF = Row number 15
15	0x0	GPIO_9_ROW_EN: Indicates whether GPIO pin# 9 is mapped to any row of keypad matrix. This bit overrides any setting done for pin# 9 in column configuration 0 = NOT_MAPPED

Bit	Reset	Description
		1 = MAPPED
14:11	0x0	GPIO_8_ROW_NUM: Mapping of GPIO pin# 8 to row number. Valid only if GPIO_8_ROW_EN is set. Indicates row number 0x0 = Row number 0 0xF = Row number 15
10	0x0	GPIO_8_ROW_EN: Indicates whether GPIO pin# 8 is mapped to any row of keypad matrix. This bit overrides any setting done for pin# 8 in column configuration 0 = NOT_MAPPED 1 = MAPPED
9:6	0x0	GPIO_7_ROW_NUM: Mapping of GPIO pin# 7 to row number. Valid only if GPIO_7_ROW_EN is set. Indicates row number 0x0 = Row number 0 0xF = Row number 15
5	0x0	GPIO_7_ROW_EN: Indicates whether GPIO pin# 7 is mapped to any row of keypad matrix. This bit overrides any setting done for pin# 7 in column configuration 0 = NOT_MAPPED 1 = MAPPED
4:1	0x0	GPIO_6_ROW_NUM: Mapping of GPIO pin# 6 to row number. Valid only if GPIO_6_ROW_EN is set. Indicates row number 0x0 = Row number 0 0xF = Row number 15
0	0x0	GPIO_6_ROW_EN: Indicates whether GPIO pin# 6 is mapped to any row of keypad matrix. This bit overrides any setting done for pin# 6 in column configuration 0 = NOT_MAPPED 1 = MAPPED

19.3.5 APBDEV_KBC_ROW_CFG2_0

This register setting takes precedence in case same GPIO pin is configured in column register also. It does the mapping between GPIO pins and rows of keypad matrix. It configures for GPIO pin# 12 to 17.

Third Row Configuration Register

Offset: 010h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
29:26	0x0	GPIO_17_ROW_NUM: Mapping of GPIO pin# 17 to row number. Valid only if GPIO_17_ROW_EN is set. Indicates row number 0x0 = Row number 0 0xF = Row number 15
25	0x0	GPIO_17_ROW_EN: Indicates whether GPIO pin# 17 is mapped to any row of keypad matrix. This bit overrides any setting done for pin# 17 in column configuration 0 = NOT_MAPPED 1 = MAPPED
24:21	0x0	GPIO_16_ROW_NUM: Mapping of GPIO pin# 16 to row number. Valid only if GPIO_16_ROW_EN is set. Indicates row number 0x0 = Row number 0 0xF = Row number 15
20	0x0	GPIO_16_ROW_EN: Indicates whether GPIO pin# 16 is mapped to any row of keypad matrix. This bit overrides any setting done for pin# 16 in column configuration 0 = NOT_MAPPED 1 = MAPPED
19:16	0x0	GPIO_15_ROW_NUM: Mapping of GPIO pin# 15 to row number. Valid only if GPIO_15_ROW_EN is set. Indicates row number 0x0 = Row number 0 0xF = Row number 15
15	0x0	GPIO_15_ROW_EN: Indicates whether GPIO pin# 15 is mapped to any row of keypad

Bit	Reset	Description
		matrix. This bit overrides any setting done for pin# 15 in column configuration 0 = NOT_MAPPED 1 = MAPPED
14:11	0x0	GPIO_14_ROW_NUM: Mapping of GPIO pin# 14 to row number. Valid only if GPIO_14_ROW_EN is set. Indicates row number 0x0 = Row number 0 0xF = Row number 15
10	0x0	GPIO_14_ROW_EN: Indicates whether GPIO pin# 14 is mapped to any row of keypad matrix. This bit overrides any setting done for pin# 14 in column configuration 0 = NOT_MAPPED 1 = MAPPED
9:6	0x0	GPIO_13_ROW_NUM: Mapping of GPIO pin# 13 to row number. Valid only if GPIO_13_ROW_EN is set. Indicates row number 0x0 = Row number 0 0xF = Row number 15
5	0x0	GPIO_13_ROW_EN: Indicates whether GPIO pin# 13 is mapped to any row of keypad matrix. This bit overrides any setting done for pin# 13 in column configuration 0 = NOT_MAPPED 1 = MAPPED
4:1	0x0	GPIO_12_ROW_NUM: Mapping of GPIO pin# 12 to row number. Valid only if GPIO_12_ROW_EN is set. Indicates row number 0x0 = Row number 0 0xF = Row number 15
0	0x0	GPIO_12_ROW_EN: Indicates whether GPIO pin# 12 is mapped to any row of keypad matrix. This bit overrides any setting done for pin# 12 in column configuration 0 = NOT_MAPPED 1 = MAPPED

19.3.6 APBDEV_KBC_ROW_CFG3_0

This register setting takes precedence in case same GPIO pin is configured in column register also. It does the mapping between GPIO pins & rows of keypad matrix. It configures for GPIO pin# 18 to 23.

Fourth Row Configuration Register

Offset: 014h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
29:26	0x0	GPIO_23_ROW_NUM: Mapping of GPIO pin# 23 to row number. Valid only if GPIO_21 is set. Indicates row number 0x0 = Row number 0 0xF = Row number 15
25	0x0	GPIO_23_ROW_EN: Indicates whether GPIO pin# 23 is mapped to any row of keypad matrix. This bit overrides any setting done for pin# 23 in column configuration 0 = NOT_MAPPED 1 = MAPPED
24:21	0x0	GPIO_22_ROW_NUM: Mapping of GPIO pin# 22 to row number. Valid only if GPIO_21 is set. Indicates row number 0x0 = Row number 0 0xF = Row number 15
20	0x0	GPIO_22_ROW_EN: Indicates whether GPIO pin# 22 is mapped to any row of keypad matrix. This bit overrides any setting done for pin# 22 in column configuration 0 = NOT_MAPPED 1 = MAPPED
19:16	0x0	GPIO_21_ROW_NUM: Mapping of GPIO pin# 21 to row number. Valid only if GPIO_21 is set. Indicates row number 0x0 = Row number 0 0xF = Row number 15
15	0x0	GPIO_21_ROW_EN: Indicates whether GPIO pin# 21 is mapped to any row of keypad matrix. This bit overrides any setting done for pin# 21 in column configuration

Bit	Reset	Description
		0 = NOT_MAPPED 1 = MAPPED
14:11	0x0	GPIO_20_ROW_NUM: Mapping of GPIO pin# 20 to row number. Valid only if GPIO_20 is set. Indicates row number 0x0 = Row number 0 0xF = Row number 15
10	0x0	GPIO_20_ROW_EN: Indicates whether GPIO pin# 20 is mapped to any row of keypad matrix. This bit overrides any setting done for pin# 20 in column configuration 0 = NOT_MAPPED 1 = MAPPED
9:6	0x0	GPIO_19_ROW_NUM: Mapping of GPIO pin# 19 to row number. Valid only if GPIO_19_ROW_EN is set. Indicates row number 0x0 = Row number 0 0xF = Row number 15
5	0x0	GPIO_19_ROW_EN: Indicates whether GPIO pin# 19 is mapped to any row of keypad matrix. This bit overrides any setting done for pin# 19 in column configuration 0 = NOT_MAPPED 1 = MAPPED
4:1	0x0	GPIO_18_ROW_NUM: Mapping of GPIO pin# 18 to row number. Valid only if GPIO_18_ROW_EN is set. Indicates row number 0x0 = Row number 0 0xF = Row number 15
0	0x0	GPIO_18_ROW_EN: Indicates whether GPIO pin# 18 is mapped to any row of keypad matrix. This bit overrides any setting done for pin# 18 in column configuration 0 = NOT_MAPPED 1 = MAPPED

19.3.7 APBDEV_KBC_COL_CFG0_0

This register does the mapping between GPIO pins & columns of keypad matrix. It configures for GPIO pin# 0 to 7.

First Column Configuration Register

Offset: 018h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:29	0x0	GPIO_7_COL_NUM: Mapping of GPIO pin# 7 to column number. Valid only if GPIO_7_COL_EN is set. Indicates row number 0x0 = Column number 0 0x7 = Column number 7
28	0x0	GPIO_7_COL_EN: Indicates whether GPIO pin# 7 is mapped to any column of keypad matrix. This bit should be set to '1' only when GPIO_7_ROW_EN in ROW_CFG1 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
27:25	0x0	GPIO_6_COL_NUM: Mapping of GPIO pin# 6 to column number. Valid only if GPIO_6_COL_EN is set. Indicates row number 0x0 = Column number 0 0x7 = Column number 7
24	0x0	GPIO_6_COL_EN: Indicates whether GPIO pin# 6 is mapped to any column of keypad matrix. This bit should be set to '1' only when GPIO_6_ROW_EN in ROW_CFG1 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
23:21	0x0	GPIO_5_COL_NUM: Mapping of GPIO pin# 5 to column number. Valid only if GPIO_5_COL_EN is set. Indicates row number 0x0 = Column number 0 0x7 = Column number 7
20	0x0	GPIO_5_COL_EN: Indicates whether GPIO pin# 5 is mapped to any column of keypad matrix. This bit should be set to '1' only when GPIO_5_ROW_EN in ROW_CFG0 is set to 0.

Bit	Reset	Description
		0 = NOT_MAPPED 1 = MAPPED
19:17	0x0	GPIO_4_COL_NUM: Mapping of GPIO pin# 4 to column number. Valid only if GPIO_4_COL_EN is set. Indicates row number 0x0 = Column number 0 0x7 = Column number 7
16	0x0	GPIO_4_COL_EN: Indicates whether GPIO pin# 4 is mapped to any column of keypad matrix. This bit should be set to '1' only when GPIO_4_ROW_EN in ROW_CFG0 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
15:13	0x0	GPIO_3_COL_NUM: Mapping of GPIO pin# 3 to column number. Valid only if GPIO_3_COL_EN is set. Indicates row number 0x0 = Column number 0 0x7 = Column number 7
12	0x0	GPIO_3_COL_EN: Indicates whether GPIO pin# 3 is mapped to any column of keypad matrix. This bit should be set to '1' only when GPIO_3_ROW_EN in ROW_CFG0 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
11:9	0x0	GPIO_2_COL_NUM: Mapping of GPIO pin# 2 to column number. Valid only if GPIO_2_COL_EN is set. Indicates row number 0x0 = Column number 0 0x7 = Column number 7
8	0x0	GPIO_2_COL_EN: Indicates whether GPIO pin# 2 is mapped to any column of keypad matrix. This bit should be set to '1' only when GPIO_2_ROW_EN in ROW_CFG0 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
7:5	0x0	GPIO_1_COL_NUM: Mapping of GPIO pin# 1 to column number. Valid only if GPIO_1_COL_EN is set. Indicates row number 0x0 = Column number 0 0x7 = Column number 7
4	0x0	GPIO_1_COL_EN: Indicates whether GPIO pin# 1 is mapped to any column of keypad matrix. This bit should be set to '1' only when GPIO_1_ROW_EN in ROW_CFG0 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
3:1	0x0	GPIO_0_COL_NUM: Mapping of GPIO pin# 0 to column number. Valid only if GPIO_0_COL_EN is set. Indicates row number 0x0 = Column number 0 0x7 = Column number 7
0	0x0	GPIO_0_COL_EN: Indicates whether GPIO pin# 0 is mapped to any column of keypad matrix. This bit should be set to '1' only when GPIO_0_ROW_EN in ROW_CFG0 is set to 0. 0 = NOT_MAPPED 1 = MAPPED

19.3.8 APBDEV_KBC_COL_CFG1_0

It does the mapping between GPIO pins & columns of keypad matrix. It configures for GPIO pin# 8 to 15.

Second Column Configuration Register

Offset: 01ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:29	0x0	GPIO_15_COL_NUM: Mapping of GPIO pin# 15 to column number. Valid only if GPIO_15_COL_EN is set. Indicates row number 0x0 = Column number 0 0x7 = Column number 7
28	0x0	GPIO_15_COL_EN: Indicates whether GPIO pin# 15 is mapped to any column of keypad matrix. This bit should be set to '1' only when GPIO_15_ROW_EN in ROW_CFG2 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
27:25	0x0	GPIO_14_COL_NUM: Mapping of GPIO pin# 14 to column number. Valid only if GPIO_14_COL_EN is set. Indicates row number 0x0 = Column number 0 0x7 = Column number 7
24	0x0	GPIO_14_COL_EN: Indicates whether GPIO pin# 14 is mapped to any column of keypad matrix. This bit should be set to '1' only when GPIO_14_ROW_EN in ROW_CFG2 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
23:21	0x0	GPIO_13_COL_NUM: Mapping of GPIO pin# 13 to column number. Valid only if GPIO_13_COL_EN is set. Indicates row number 0x0 = Column number 0 0x7 = Column number 7
20	0x0	GPIO_13_COL_EN: Indicates whether GPIO pin# 13 is mapped to any column of keypad matrix. This bit should be set to '1' only when GPIO_13_ROW_EN in ROW_CFG2 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
19:17	0x0	GPIO_12_COL_NUM: Mapping of GPIO pin# 12 to column number. Valid only if GPIO_12_COL_EN is set. Indicates row number 0x0 = Column number 0 0x7 = Column number 7
16	0x0	GPIO_12_COL_EN: Indicates whether GPIO pin# 12 is mapped to any column of keypad matrix. This bit should be set to '1' only when GPIO_12_ROW_EN in ROW_CFG2 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
15:13	0x0	GPIO_11_COL_NUM: Mapping of GPIO pin# 11 to column number. Valid only if GPIO_11_COL_EN is set. Indicates row number 0x0 = Column number 0 0x7 = Column number 7
12	0x0	GPIO_11_COL_EN: Indicates whether GPIO pin# 11 is mapped to any column of keypad matrix. This bit should be set to '1' only when GPIO_11_ROW_EN in ROW_CFG1 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
11:9	0x0	GPIO_10_COL_NUM: Mapping of GPIO pin# 10 to column number. Valid only if GPIO_10_COL_EN is set. Indicates row number 0x0 = Column number 0 0x7 = Column number 7
8	0x0	GPIO_10_COL_EN: Indicates whether GPIO pin# 10 is mapped to any column of keypad matrix. This bit should be set to '1' only when GPIO_10_ROW_EN in ROW_CFG1 is set to 0.

Bit	Reset	Description
		0 = NOT_MAPPED 1 = MAPPED
7:5	0x0	GPIO_9_COL_NUM: Mapping of GPIO pin# 9 to column number. Valid only if GPIO_9_COL_EN is set. Indicates row number 0x0 = Column number 0 0x7 = Column number 7
4	0x0	GPIO_9_COL_EN: Indicates whether GPIO pin# 9 is mapped to any column of keypad matrix. This bit should be set to '1' only when GPIO_9_ROW_EN in ROW_CFG1 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
3:1	0x0	GPIO_8_COL_NUM: Mapping of GPIO pin# 8 to column number. Valid only if GPIO_8_COL_EN is set. Indicates row number 0x0 = Column number 0 0x7 = Column number 7
0	0x0	GPIO_8_COL_EN: Indicates whether GPIO pin# 8 is mapped to any column of keypad matrix. This bit should be set to '1' only when GPIO_8_ROW_EN in ROW_CFG1 is set to 0. 0 = NOT_MAPPED 1 = MAPPED

19.3.9 APBDEV_KBC_COL_CFG2_0

It does the mapping between GPIO pins & columns of keypad matrix. It configures for GPIO pin# 16 to 23.

Third Column Configuration Register

Offset: 020h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:29	0x0	GPIO_23_COL_NUM: Mapping of GPIO pin# 23 to column number. Valid only if GPIO_23 is set. Indicates row number 0x0 = Column number 0 0x7 = Column number 7
28	0x0	GPIO_23_COL_EN: Indicates whether GPIO pin# 23 is mapped to any column of keypad matrix. This bit should be set to '1' only when GPIO_22 in ROW_CFG3 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
27:25	0x0	GPIO_22_COL_NUM: Mapping of GPIO pin# 22 to column number. Valid only if GPIO_22 is set. Indicates row number 0x0 = Column number 0 0x7 = Column number 7
24	0x0	GPIO_22_COL_EN: Indicates whether GPIO pin# 22 is mapped to any column of keypad matrix. This bit should be set to '1' only when GPIO_22 in ROW_CFG3 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
23:21	0x0	GPIO_21_COL_NUM: Mapping of GPIO pin# 21 to column number. Valid only if GPIO_21 is set. Indicates row number 0x0 = Column number 0 0x7 = Column number 7
20	0x0	GPIO_21_COL_EN: Indicates whether GPIO pin# 21 is mapped to any column of keypad matrix. This bit should be set to '1' only when GPIO_21 in ROW_CFG3 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
19:17	0x0	GPIO_20_COL_NUM: Mapping of GPIO pin# 20 to column number. Valid only if GPIO_20 is set. Indicates row number 0x0 = Column number 0 0x7 = Column number 7
16	0x0	GPIO_20_COL_EN: Indicates whether GPIO pin# 20 is mapped to any column of keypad matrix. This bit should be set to '1' only when GPIO_20 in ROW_CFG3 is set to 0. 0 = NOT_MAPPED

Bit	Reset	Description
		1 = MAPPED
15:13	0x0	GPIO_19_COL_NUM: Mapping of GPIO pin# 19 to column number. Valid only if GPIO_19_COL_EN is set. Indicates row number 0x0 = Column number 0 0x7 = Column number 7
12	0x0	GPIO_19_COL_EN: Indicates whether GPIO pin# 19 is mapped to any column of keypad matrix. This bit should be set to '1' only when GPIO_19_ROW_EN in ROW_CFG3 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
11:9	0x0	GPIO_18_COL_NUM: Mapping of GPIO pin# 18 to column number. Valid only if GPIO_18_COL_EN is set. Indicates row number 0x0 = Column number 0 0x7 = Column number 7
8	0x0	GPIO_18_COL_EN: Indicates whether GPIO pin# 18 is mapped to any column of keypad matrix. This bit should be set to '1' only when GPIO_18_ROW_EN in ROW_CFG3 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
7:5	0x0	GPIO_17_COL_NUM: Mapping of GPIO pin# 17 to column number. Valid only if GPIO_17_COL_EN is set. Indicates row number 0x0 = Column number 0 0x7 = Column number 7
4	0x0	GPIO_17_COL_EN: Indicates whether GPIO pin# 17 is mapped to any column of keypad matrix. This bit should be set to '1' only when GPIO_17_ROW_EN in ROW_CFG2 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
3:1	0x0	GPIO_16_COL_NUM: Mapping of GPIO pin# 16 to column number. Valid only if GPIO_16_COL_EN is set. Indicates row number 0x0 = Column number 0 0x7 = Column number 7
0	0x0	GPIO_16_COL_EN: Indicates whether GPIO pin# 16 is mapped to any column of keypad matrix. This bit should be set to '1' only when GPIO_16_ROW_EN in ROW_CFG2 is set to 0. 0 = NOT_MAPPED 1 = MAPPED

19.3.10 APBDEV_KBC_TO_CNT_0

This register holds the time-out count value. In CP state, if the corresponding counter crosses this value KBC goes back to WuKP (Wake-up on Key-press state).

Continuous Polling (CP) State Time-out Count Register

Offset: 024h | Read/Write: R/W | Reset: 0b00100111000100000000

Bit	Reset	Description
19:0	0x27100	TO_CNT_VAL: Time-out count value. The default value is 5 seconds. The value should be calculated for a 32 KHz clock.

19.3.11 APBDEV_KBC_INIT_DLY_0

This register holds the initial delay for scan when KBC goes from WuKP mode to CP mode.

Offset: 028h | Read/Write: R/W | Reset: 0b00000000010000000000

Bit	Reset	Description
19:0	0x400	INIT_DLY_VAL: Initial delay value. The default value is 32.25 milliseconds. The value should be calculated for a 32 KHz clock.

19.3.12 APBDEV_KBC_RPT_DLY_0

This register holds the repeat scan delay. This is the delay between two successive scans in CP mode.

Offset: 02ch | Read/Write: R/W | Reset: 0b00000000010000000000

Bit	Reset	Description
19:0	0x400	RPT_DLY_VAL: delay value. The default value is 32.25 milliseconds. The value should be calculated for a 32 KHz clock.

19.3.13 APBDEV_KBC_KP_ENT0_0

This register holds the key numbers of the keys which have been pressed.

It holds the first four entries

NOTE: Both the entry registers should be read consecutively. A read of first register changes the FIFO read pointer

First Key-press Entries Register

Offset: 030h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	KP_NEW_ENT3: Indicates whether fourth entry is valid or not 0x0 Entry not valid 0x1 Valid entry
30:27	X	KP_ROW_NUM_ENT3: Row number for fourth key. 0x0 = Row number 0 0xF = Row number 15
26:24	X	KP_COL_NUM_ENT3: Column number for fourth key. 0x0 = Column number 0 0x7 = Column number 7
23	X	KP_NEW_ENT2: Indicates whether third entry is valid or not 0x0 Entry not valid 0x1 Valid entry
22:19	X	KP_ROW_NUM_ENT2: Row number for third key. 0x0 = Row number 0 0xF = Row number 15
18:16	X	KP_COL_NUM_ENT2: Column number for third key. 0x0 = Column number 0 0x7 = Column number 7
15	X	KP_NEW_ENT1: Indicates whether second entry is valid or not 0x0 Entry not valid 0x1 Valid entry
14:11	X	KP_ROW_NUM_ENT1: Row number for second key. 0x0 = Row number 0 0xF = Row number 15
10:8	X	KP_COL_NUM_ENT1: Column number for second key. 0x0 = Column number 0 0x7 = Column number 7

Bit	Reset	Description
7	X	KP_NEW_ENT0: Indicates whether first entry is valid or not 0x0 Entry not valid 0x1 Valid entry
6:3	X	KP_ROW_NUM_ENT0: Row number for first key. 0x0 = Row number 0 0xF = Row number 15
2:0	X	KP_COL_NUM_ENT0: Column number for first key. 0x0 = Column number 0 0x7 = Column number 7

19.3.14 APBDEV_KBC_KP_ENT1_0

This register holds the key numbers of the keys which have been pressed.

It holds the last four entries (out of eight entries)

NOTE: Both the entry registers should be read consecutively. A read of first register changes the FIFO read pointer

Second Key-press Entries Register

Offset: 034h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	KP_NEW_ENT7: Indicates whether eight entry is valid or not 0x0 Entry not valid 0x1 Valid entry
30:27	X	KP_ROW_NUM_ENT7: Row number for eight key. 0x0 = Row number 0 0xF = Row number 15
26:24	X	KP_COL_NUM_ENT7: Column number for eight key. 0x0 = Column number 0 0x7 = Column number 7
23	X	KP_NEW_ENT6: Indicates whether seventh entry is valid or not 0x0 Entry not valid 0x1 Valid entry
22:19	X	KP_ROW_NUM_ENT6: Row number for seventh key. 0x0 = Row number 0 0xF = Row number 15
18:16	X	KP_COL_NUM_ENT6: Column number for seventh key. 0x0 = Column number 0 0x7 = Column number 7
15	X	KP_NEW_ENT5: Indicates whether sixth entry is valid or not 0x0 Entry not valid 0x1 Valid entry
14:11	X	KP_ROW_NUM_ENT5: Row number for sixth key. 0x0 = Row number 0 0xF = Row number 15
10:8	X	KP_COL_NUM_ENT5: Column number for sixth key. 0x0 = Column number 0 0x7 = Column number 7
7	X	KP_NEW_ENT4: Indicates whether fifth entry is valid or not 0x0 Entry not valid 0x1 Valid entry
6:3	X	KP_ROW_NUM_ENT4: Row number for fifth key. 0x0 = Row number 0 0xF = Row number 15
2:0	X	KP_COL_NUM_ENT4: Column number for fifth key. 0x0 = Column number 0 0x7 = Column number 7

19.3.15 APBDEV_KBC_ROW0_MASK_0

Offset: 038h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	KBC_ROW0_COL7_MASK_ENABLE: Disable row0 col7 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair

Bit	Reset	Description
		0 = ENABLE 1 = DISABLE
6	0x0	KBC_ROW0_COL6_MASK_ENABLE: Disable row0 col6 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
5	0x0	KBC_ROW0_COL5_MASK_ENABLE: Disable row0 col5 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
4	0x0	KBC_ROW0_COL4_MASK_ENABLE: Disable row0 col4 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
3	0x0	KBC_ROW0_COL3_MASK_ENABLE: Disable row0 col3 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
2	0x0	KBC_ROW0_COL2_MASK_ENABLE: Disable row0 col2 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
1	0x0	KBC_ROW0_COL1_MASK_ENABLE: Disable row0 col1 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
0	0x0	KBC_ROW0_COL0_MASK_ENABLE: Disable row0 col0 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

19.3.16 APBDEV_KBC_ROW1_MASK_0

Offset: 03ch | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	KBC_ROW1_COL7_MASK_ENABLE: Disable row1 col7 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
6	0x0	KBC_ROW1_COL6_MASK_ENABLE: Disable row1 col6 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
5	0x0	KBC_ROW1_COL5_MASK_ENABLE: Disable row1 col5 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
4	0x0	KBC_ROW1_COL4_MASK_ENABLE: Disable row1 col4 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

Bit	Reset	Description
3	0x0	KBC_ROW1_COL3_MASK_ENABLE: Disable row1 col3 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
2	0x0	KBC_ROW1_COL2_MASK_ENABLE: Disable row1 col2 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
1	0x0	KBC_ROW1_COL1_MASK_ENABLE: Disable row1 col1 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
0	0x0	KBC_ROW1_COL0_MASK_ENABLE: Disable row1 col0 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

19.3.17 APBDEV_KBC_ROW2_MASK_0

Offset: 040h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	KBC_ROW2_COL7_MASK_ENABLE: Disable row2 col7 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
6	0x0	KBC_ROW2_COL6_MASK_ENABLE: Disable row2 col6 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
5	0x0	KBC_ROW2_COL5_MASK_ENABLE: Disable row2 col5 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
4	0x0	KBC_ROW2_COL4_MASK_ENABLE: Disable row2 col4 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
3	0x0	KBC_ROW2_COL3_MASK_ENABLE: Disable row2 col3 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
2	0x0	KBC_ROW2_COL2_MASK_ENABLE: Disable row2 col2 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
1	0x0	KBC_ROW2_COL1_MASK_ENABLE: Disable row2 col1 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
0	0x0	KBC_ROW2_COL0_MASK_ENABLE: Disable row2 col0 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE

Bit	Reset	Description
		1 = DISABLE

19.3.18 APBDEV_KBC_ROW3_MASK_0

Offset: 044h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	KBC_ROW3_COL7_MASK_ENABLE: Disable row3 col7 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
6	0x0	KBC_ROW3_COL6_MASK_ENABLE: Disable row3 col6 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
5	0x0	KBC_ROW3_COL5_MASK_ENABLE: Disable row3 col5 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
4	0x0	KBC_ROW3_COL4_MASK_ENABLE: Disable row3 col4 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
3	0x0	KBC_ROW3_COL3_MASK_ENABLE: Disable row3 col3 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
2	0x0	KBC_ROW3_COL2_MASK_ENABLE: Disable row3 col2 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
1	0x0	KBC_ROW3_COL1_MASK_ENABLE: Disable row3 col1 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
0	0x0	KBC_ROW3_COL0_MASK_ENABLE: Disable row3 col0 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

19.3.19 APBDEV_KBC_ROW4_MASK_0

Offset: 048h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	KBC_ROW4_COL7_MASK_ENABLE: Disable row4 col7 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
6	0x0	KBC_ROW4_COL6_MASK_ENABLE: Disable row4 col6 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

Bit	Reset	Description
5	0x0	KBC_ROW4_COL5_MASK_ENABLE: Disable row4 col5 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
4	0x0	KBC_ROW4_COL4_MASK_ENABLE: Disable row4 col4 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
3	0x0	KBC_ROW4_COL3_MASK_ENABLE: Disable row4 col3 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
2	0x0	KBC_ROW4_COL2_MASK_ENABLE: Disable row4 col2 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
1	0x0	KBC_ROW4_COL1_MASK_ENABLE: Disable row4 col1 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
0	0x0	KBC_ROW4_COL0_MASK_ENABLE: Disable row4 col0 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

19.3.20 APBDEV_KBC_ROW5_MASK_0

Offset: 04ch | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	KBC_ROW5_COL7_MASK_ENABLE: Disable row5 col7 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
6	0x0	KBC_ROW5_COL6_MASK_ENABLE: Disable row5 col6 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
5	0x0	KBC_ROW5_COL5_MASK_ENABLE: Disable row5 col5 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
4	0x0	KBC_ROW5_COL4_MASK_ENABLE: Disable row5 col4 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
3	0x0	KBC_ROW5_COL3_MASK_ENABLE: Disable row5 col3 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
2	0x0	KBC_ROW5_COL2_MASK_ENABLE: Disable row5 col2 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE

Bit	Reset	Description
		1 = DISABLE
1	0x0	KBC_ROW5_COL1_MASK_ENABLE: Disable row5 col1 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
0	0x0	KBC_ROW5_COL0_MASK_ENABLE: Disable row5 col0 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

19.3.21 APBDEV_KBC_ROW6_MASK_0

Offset: 050h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	KBC_ROW6_COL7_MASK_ENABLE: Disable row6 col7 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
6	0x0	KBC_ROW6_COL6_MASK_ENABLE: Disable row6 col6 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
5	0x0	KBC_ROW6_COL5_MASK_ENABLE: Disable row6 col5 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
4	0x0	KBC_ROW6_COL4_MASK_ENABLE: Disable row6 col4 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
3	0x0	KBC_ROW6_COL3_MASK_ENABLE: Disable row6 col3 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
2	0x0	KBC_ROW6_COL2_MASK_ENABLE: Disable row6 col2 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
1	0x0	KBC_ROW6_COL1_MASK_ENABLE: Disable row6 col1 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
0	0x0	KBC_ROW6_COL0_MASK_ENABLE: Disable row6 col0 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

19.3.22 APBDEV_KBC_ROW7_MASK_0

Offset: 054h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	KBC_ROW7_COL7_MASK_ENABLE: Disable row7 col7 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
6	0x0	KBC_ROW7_COL6_MASK_ENABLE: Disable row7 col6 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
5	0x0	KBC_ROW7_COL5_MASK_ENABLE: Disable row7 col5 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
4	0x0	KBC_ROW7_COL4_MASK_ENABLE: Disable row7 col4 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
3	0x0	KBC_ROW7_COL3_MASK_ENABLE: Disable row7 col3 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
2	0x0	KBC_ROW7_COL2_MASK_ENABLE: Disable row7 col2 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
1	0x0	KBC_ROW7_COL1_MASK_ENABLE: Disable row7 col1 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
0	0x0	KBC_ROW7_COL0_MASK_ENABLE: Disable row7 col0 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

19.3.23 APBDEV_KBC_ROW8_MASK_0

Offset: 058h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	KBC_ROW8_COL7_MASK_ENABLE: Disable row8 col7 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
6	0x0	KBC_ROW8_COL6_MASK_ENABLE: Disable row8 col6 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
5	0x0	KBC_ROW8_COL5_MASK_ENABLE: Disable row8 col5 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

Bit	Reset	Description
4	0x0	KBC_ROW8_COL4_MASK_ENABLE: Disable row8 col4 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
3	0x0	KBC_ROW8_COL3_MASK_ENABLE: Disable row8 col3 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
2	0x0	KBC_ROW8_COL2_MASK_ENABLE: Disable row8 col2 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
1	0x0	KBC_ROW8_COL1_MASK_ENABLE: Disable row8 col1 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
0	0x0	KBC_ROW8_COL0_MASK_ENABLE: Disable row8 col0 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

19.3.24 APBDEV_KBC_ROW9_MASK_0

Offset: 05ch | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	KBC_ROW9_COL7_MASK_ENABLE: Disable row9 col7 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
6	0x0	KBC_ROW9_COL6_MASK_ENABLE: Disable row9 col6 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
5	0x0	KBC_ROW9_COL5_MASK_ENABLE: Disable row9 col5 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
4	0x0	KBC_ROW9_COL4_MASK_ENABLE: Disable row9 col4 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
3	0x0	KBC_ROW9_COL3_MASK_ENABLE: Disable row9 col3 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
2	0x0	KBC_ROW9_COL2_MASK_ENABLE: Disable row9 col2 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
1	0x0	KBC_ROW9_COL1_MASK_ENABLE: Disable row9 col1 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE

Bit	Reset	Description
		1 = DISABLE
0	0x0	KBC_ROW9_COL0_MASK_ENABLE: Disable row9 col0 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

19.3.25 APBDEV_KBC_ROW10_MASK_0

Offset: 060h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	KBC_ROW10_COL7_MASK_ENABLE: Disable row10 col7 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
6	0x0	KBC_ROW10_COL6_MASK_ENABLE: Disable row10 col6 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
5	0x0	KBC_ROW10_COL5_MASK_ENABLE: Disable row10 col5 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
4	0x0	KBC_ROW10_COL4_MASK_ENABLE: Disable row10 col4 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
3	0x0	KBC_ROW10_COL3_MASK_ENABLE: Disable row10 col3 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
2	0x0	KBC_ROW10_COL2_MASK_ENABLE: Disable row10 col2 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
1	0x0	KBC_ROW10_COL1_MASK_ENABLE: Disable row10 col1 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
0	0x0	KBC_ROW10_COL0_MASK_ENABLE: Disable row10 col0 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

19.3.26 APBDEV_KBC_ROW11_MASK_0

Offset: 064h | Read/Write: R/W | Secure: Unprotected | Reset: 0b00000000 | Default: 0000.0000

Bit	Reset	Description
7	0x0	KBC_ROW11_COL7_MASK_ENABLE: Disable row11 col7 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

Bit	Reset	Description
6	0x0	KBC_ROW11_COL6_MASK_ENABLE: Disable row11 col6 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
5	0x0	KBC_ROW11_COL5_MASK_ENABLE: Disable row11 col5 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
4	0x0	KBC_ROW11_COL4_MASK_ENABLE: Disable row11 col4 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
3	0x0	KBC_ROW11_COL3_MASK_ENABLE: Disable row11 col3 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
2	0x0	KBC_ROW11_COL2_MASK_ENABLE: Disable row11 col2 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
1	0x0	KBC_ROW11_COL1_MASK_ENABLE: Disable row11 col1 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
0	0x0	KBC_ROW11_COL0_MASK_ENABLE: Disable row11 col0 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

19.3.27 APBDEV_KBC_ROW12_MASK_0

Offset: 068h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	KBC_ROW12_COL7_MASK_ENABLE: Disable row12 col7 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
6	0x0	KBC_ROW12_COL6_MASK_ENABLE: Disable row12 col6 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
5	0x0	KBC_ROW12_COL5_MASK_ENABLE: Disable row12 col5 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
4	0x0	KBC_ROW12_COL4_MASK_ENABLE: Disable row12 col4 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
3	0x0	KBC_ROW12_COL3_MASK_ENABLE: Disable row12 col3 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE

Bit	Reset	Description
		1 = DISABLE
2	0x0	KBC_ROW12_COL2_MASK_ENABLE: Disable row12 col2 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
1	0x0	KBC_ROW12_COL1_MASK_ENABLE: Disable row12 col1 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
0	0x0	KBC_ROW12_COL0_MASK_ENABLE: Disable row12 col0 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

19.3.28 APBDEV_KBC_ROW13_MASK_0

Offset: 06ch | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	KBC_ROW13_COL7_MASK_ENABLE: Disable row13 col7 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
6	0x0	KBC_ROW13_COL6_MASK_ENABLE: Disable row13 col6 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
5	0x0	KBC_ROW13_COL5_MASK_ENABLE: Disable row13 col5 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
4	0x0	KBC_ROW13_COL4_MASK_ENABLE: Disable row13 col4 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
3	0x0	KBC_ROW13_COL3_MASK_ENABLE: Disable row13 col3 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
2	0x0	KBC_ROW13_COL2_MASK_ENABLE: Disable row13 col2 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
1	0x0	KBC_ROW13_COL1_MASK_ENABLE: Disable row13 col1 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
0	0x0	KBC_ROW13_COL0_MASK_ENABLE: Disable row13 col0 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

19.3.29 APBDEV_KBC_ROW14_MASK_0

Offset: 070h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	KBC_ROW14_COL7_MASK_ENABLE: Disable row14 col7 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
6	0x0	KBC_ROW14_COL6_MASK_ENABLE: Disable row14 col6 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
5	0x0	KBC_ROW14_COL5_MASK_ENABLE: Disable row14 col5 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
4	0x0	KBC_ROW14_COL4_MASK_ENABLE: Disable row14 col4 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
3	0x0	KBC_ROW14_COL3_MASK_ENABLE: Disable row14 col3 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
2	0x0	KBC_ROW14_COL2_MASK_ENABLE: Disable row14 col2 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
1	0x0	KBC_ROW14_COL1_MASK_ENABLE: Disable row14 col1 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
0	0x0	KBC_ROW14_COL0_MASK_ENABLE: Disable row14 col0 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

19.3.30 APBDEV_KBC_ROW15_MASK_0

Offset: 074h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	KBC_ROW15_COL7_MASK_ENABLE: Disable row15 col7 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
6	0x0	KBC_ROW15_COL6_MASK_ENABLE: Disable row15 col6 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
5	0x0	KBC_ROW15_COL5_MASK_ENABLE: Disable row15 col5 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

Bit	Reset	Description
4	0x0	KBC_ROW15_COL4_MASK_ENABLE: Disable row15 col4 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
3	0x0	KBC_ROW15_COL3_MASK_ENABLE: Disable row15 col3 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
2	0x0	KBC_ROW15_COL2_MASK_ENABLE: Disable row15 col2 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
1	0x0	KBC_ROW15_COL1_MASK_ENABLE: Disable row15 col1 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
0	0x0	KBC_ROW15_COL0_MASK_ENABLE: Disable row15 col0 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

20.0 PWFM CONTROLLER

The Pulse Width Frequency Modulator (PWFM) controller is a four channel frequency divider whose pulse width varies. Each channel has a programmable frequency divider and a programmable pulse width generator.

Frequency division is a 13 bit programmable value and pulse division is an 8 bit value.

The PWFM runs off a device clock which is programmed in the Clock and Reset controller, and can be any frequency up to the device clock maximum speed of 48 MHz.

The device clock frequency is then divided by 256, before subdividing it further based on the programmable value locally.

There is an APB interface which interfaces the register logic to the APB bus.

The Tegra® 2 Series PWFM contains four independently programmable pulse width modulators. Each generated pulse has an n/256 duty cycle. PWM signals are useful for LCD contrast and brightness control, VCO-generated clocks and other analog voltage references where high precision is not required.

20.1 Functionality

There are four PWFM controllers on four individual pins on the chip. The pin mux corresponding to these are SDC primary pin groups. For the four PWFM controllers there are four corresponding registers PWM_CSR0, PWM_CSR1, PWM_CSR2 and PWM_CSR3. Each PWM must be enabled (bit[31]) to be operational.

Pulse Width [23:16] determines the output pulse width and must be programmed appropriately. Frequency Divider [12:0] determines the divided clock.

The output is generated from the dividers, with each divider generating one output.

20.2 PWFM Registers

20.2.1 PWM_CONTROLLER_PWM_CSR_0_0

PWM Output-0 Configuration Control Register

Offset: 000h | Read/Write: R/W | Reset: 0b0000000000000000xxx0000000000000

Bit	Reset	Description
31	0x0	ENB: Enable Pulse width modulator 0 = DISABLE 1 = ENABLE
30:16	0x0	PWM_0: pulse width that needs to be programmed. 0=Always low 1=1/256 Pulse high 2=2/256 Pulse High N=N/256 Pulse high
12:0	0x0	PFM_0: Frequency divider that needs to be Programmed.

20.2.2 PWM_CONTROLLER_PWM_CSR_1_0

PWM Output-1 Configuration Control Register

Offset: 010h | Read/Write: R/W | Reset: 0b0000000000000000xxx0000000000000

Bit	Reset	Description
31	0x0	ENB: Enable pulse width modulator 0 = DISABLE 1 = ENABLE
30:16	0x0	PWM_0: pulse width that needs to be programmed 0=Always low 1=1/256 Pulse high 2=2/256 Pulse High ------- N=N/256 Pulse high
12:0	0x0	PFM_1: Frequency divider that needs to be Programmed.

20.2.3 PWM_CONTROLLER_PWM_CSR_2_0

PWM Output-2 Configuration Control Register

Offset: 020h | Read/Write: R/W | Reset: 0b0000000000000000xxx0000000000000

Bit	Reset	Description
31	0x0	ENB: Enable pulse width modulator 0 = DISABLE 1 = ENABLE
30:16	0x0	PWM_0: Pulse Width that needs to be programmed 0=Always low 1=1/256 Pulse high 2=2/256 Pulse High ------ N=N/256 Pulse high
12:0	0x0	PFM_2: Frequency divider that needs to be Programmed

20.2.4 PWM_CONTROLLER_PWM_CSR_3_0

PWM Output-3 Configuration Control Register

Offset: 030h | Read/Write: R/W | Reset: 0b0000000000000000xxx0000000000000

Bit	Reset	Description
31	0x0	ENB: Enable pulse width modulator 0 = DISABLE 1 = ENABLE
30:16	0x0	PWM_0: pulse width that needs to be programmed 0=Always low 1=1/256 Pulse high 2=2/256 Pulse High ------- N=N/256 Pulse high
12:0	0x0	PFM_3: Frequency divider that needs to be Programmed.

21.0 I2C CONTROLLER

The I2C controller (I2C) implements an I^2C 2.1 specification-compliant I2C master and a slave controller. I2C supports multiple masters and slaves. It also supports fast mode (400 KHz operation) and high speed mode (3.4 MHz operation). Tegra® 2 Processor has three instances of this controller. All three instances present identical I2C master functionality. The power I2C does not enable I2C slave functionality while the other two support slave functionality as well.

I2C supports DMA over APB bus in addition to single byte transfers. I2C also supports packet mode transfers where the data to be transferred is encapsulated in a predefined packet format as payload and sent to I2C over APB bus via DMA. The header of the packet specifies the type of operation to be performed, the size and other parameters (described in detail in subsequent sections).

Features

- Supports standard and fast mode of operation (0-400 KHz) as well as high speed mode (3.4 MHz).

  Note: The high speed mode of operation is not I2C 2.1 specification-compliant (requires a current source pull up to be implemented).

- Independent Master Controller and Slave Controller with separate register-based host interface
- Master supports clock stretching by the slave
- Supports one to eight-byte burst data transfers
- Both master and slave support transactions with 7-bit or 10-bit addressing
- Master is capable of data transfers to/from two or more slaves consecutively with a repeated-start condition
- Fully programmable 7-bit or 10-bit address for the slave
- Supports general call addressing
- Supports Recognition and Transfer of data to peripherals that do not send an acknowledge
- Slave can also be configured not to acknowledge
- Master supports packet based DMA

Clocking

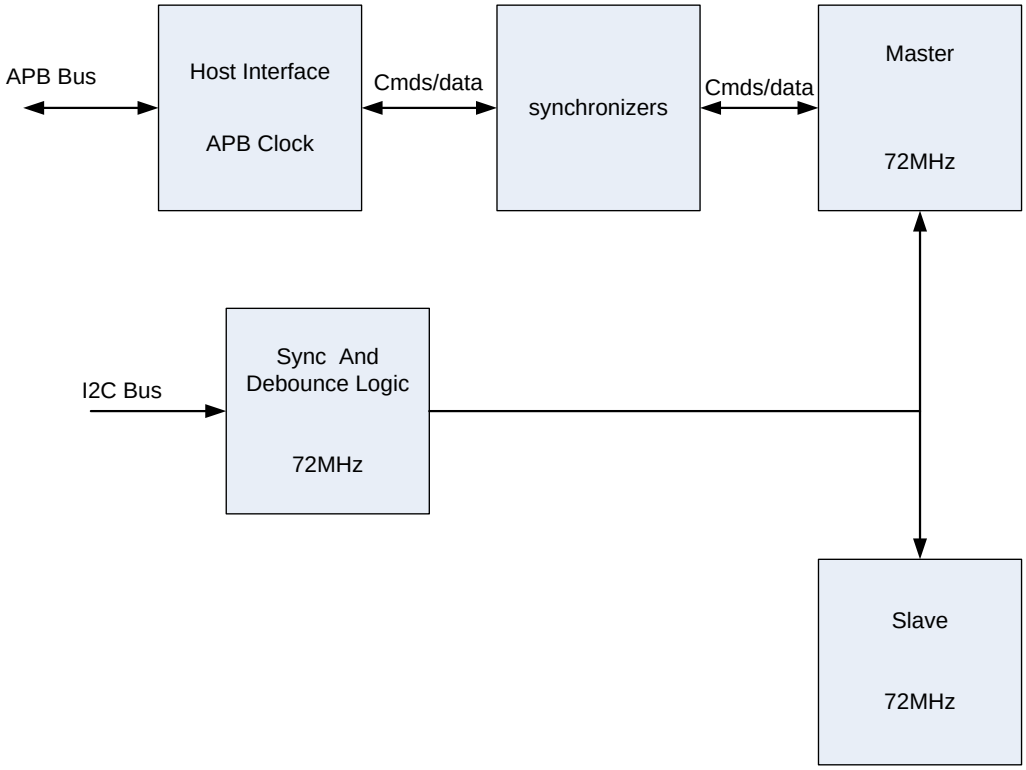
I2C has two clock domains:

- Host interface runs on APB clock. APB clock is derived from sys_clk and can be 1.0, ½, 1/3 or ¼ times the sys_clk.
- I2C interface controller runs on a fixed frequency clock running at 72MHz. The 72MHz clock is derived from PLLP. Even though we can mux between PLLP, PLLC, PLLM, and OSC clocks, PLLP is always selected and used.

21.1 Functionality

The I^2C controller can work as both a Master and a Slave. The Master can address the internal slave (for basic testing) or an external 7-bit or 10-bit addressed Slave device. The Master can be programmed for either a single slave transaction or a two-slave transaction. The two-slave transaction is generally useful in a random read from an external device. When reading from an external device, the random read-address needs to be set with an initial dummy-write to the device, followed by a repeated-start and a read transaction to the same device. The internal Slave address can be programmed to be either a 7-bit or a 10-bit address. Figure 38 below shows the I2C controller in both the master and slave modes.

Figure 38 I2C Controller Block Diagram



21.2 Interfaces

A typical peripheral controller might require the following set of steps to communicate or control an external peripheral:

1. Setup the registers
2. Program the ‘go’ bit to start the interaction with external peripheral
3. Peripheral controller collects the response from peripheral and interrupts software
4. Software reads the response

21.2.1 Packet Flow

The flow is a transport path that can be uniquely identified in the system. It is a virtual path or channel that carries a particular type of packets from a source to a destination. A flow carries logical information and is not dependent on the physical implementation.

Information exchange between hardware peripheral controller and software as described in section 21.2 can be classified into:

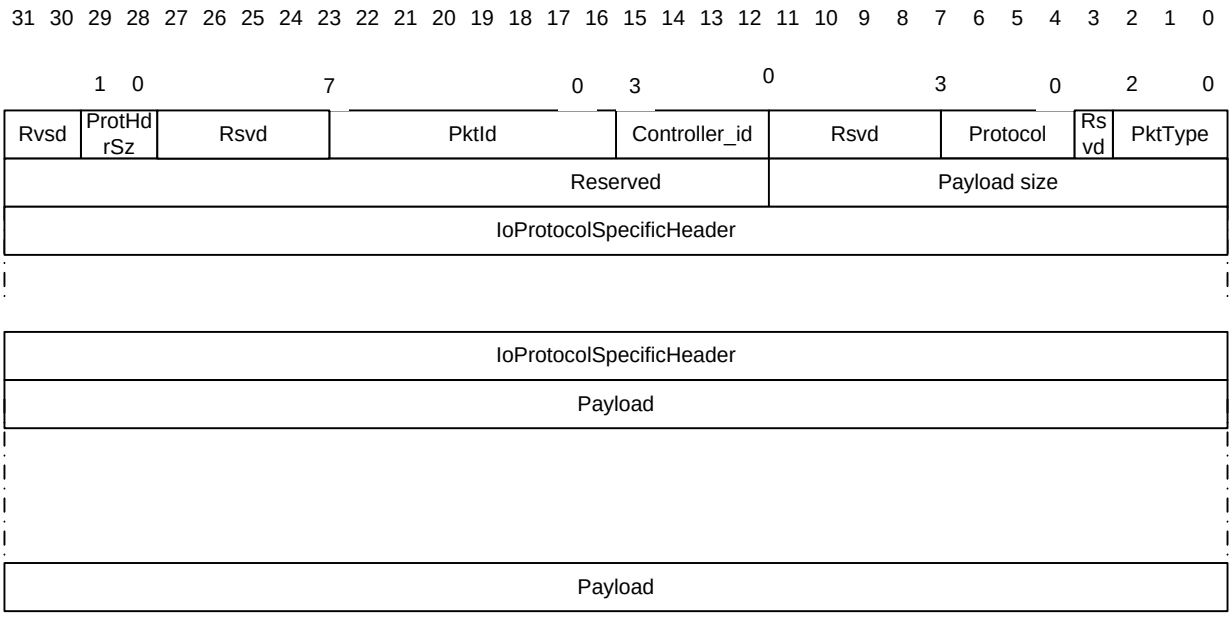
- ‘Request’ flow which can represent the register writes
- ‘Response’ flow for peripheral controller responses sent back to software, and
- ‘Interrupt’ flow for out-of-band handshake between hardware and software

Packets within a flow can be exchanged in a manner that requires full intervention from CPU (PIO mode) or least intervention from CPU (DMA mode). The IO Packet described below is a structure that can be used to represent various flows in the system.

21.2.2 IO Packet Format

An IO Packet consists of header and payload. Header consists of two parts: Two word of generic header, and 1-4 words of protocol specific header.

Figure 39 IO Packet



21.2.2.1 IO Packet Word 0

Word 0 of the IO packet contains the Generic Header Word 0. The contents of generic header word 0 are described in the table below

Table 62. Generic IO 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
27:24	Reserved	Reserved
23:16	PktId	Packet identifier
15:12	Controller ID	There might be multiple controllers supporting any format. For example there might be 4 I2C controllers. Controller ID field specifies the controller for which this packet is destined.
11:8	Reserved	Reserved
7:4	Protocol	The protocol is indicated in this field 0 = reserved 1= I2c 2-15 = reserved
3:3	Reserved	Reserved

Bits	Field Value	Description
2:0	PktType	Packet type information 0 = request 1 = response 2 = interrupt 3 = stop 4-7 = reserved

21.2.2.2 IO Packet Word 1

Word 1 of the IO packet contains the Generic Header Word 1. The contents of generic header word 1 are described in the table below.

Table 63. Generic IO Header Word 1

Bits	Field Value	Description
31:12	Reserved	Reserved
11:0	PayloadSize	Payload size in bytes

21.2.3 Transferring Packets

Packets can be transferred using PIO or DMA modes. A peripheral controller can choose to implement one or both modes. Header is a minimum of three words with the first two words reserved for packet information and the third word reserved for protocol specific requirements. A minimum of four words is needed to transfer one byte to the external peripheral.

21.2.4 IO Protocol Specific Header

Depending on the implementation of the peripheral controller and the interface protocol, the header words can be defined. No restriction is placed on the organization of the fields or the number of protocol specific words (a minimum of one word and a maximum of four words) a particular implementation chooses.

I2C has one word of Protocol Specific Header. The contents of I2C protocol specific header are described in the table below.

Table 64. IO Protocol Specific Header

Bit	Name	Description
31:23	Reserved	Reserved
22	HS_MODE	Enable high speed mode (3.4MHz operation)
21	CONTINUE_ON_NACK	Enable mode to handle devices that do not generate ACK upon the Reception of a byte.
20	SEND_START_BYTE	1 = send a start byte at the beginning of the transaction
19	READ/WRITE	1= READ
18	Address mode	1=10 bit mode 0 = 7 bit mode
17	IE	Generate interrupt upon packet completion
16	REPEAT_START/STOP	1 indicates to put a repeat start condition on the bus(to continue transaction) 0 indicates to put a stop condition on the bus
15	Reserved	Reserved
14:12	HS_MASTER_ADDR	High Speed mode Master code
11:10	Reserved	Reserved
9:0	SLAVE_ADDR	Slave address

21.3 Programming Guidelines

21.3.1 I2C Frequency Divisor Register

The FREQUENCY DIVISOR register (CLK_SOURCE_I2C register, whose address is: 0x600060A4) must be programmed as a function of the CLK_SOURCE Selected for I2C as follows:

```
I2C_CLK = CLK_SOURCE.I2C / (8 * I2C FREQUENCY DIVISOR)
```

The I^2C bus specification defines the minimum low period for the I2C_CLK as 4.7µs in standard mode and 1.3µs in fast mode. Because of this, the maximum I2C_CLK frequency in the standard mode can be 100 KHz but in fast mode, it is limited to 348 KHz, assuming I2C_CLK as rise and fall delays of 300ns per the I^2C specification.

The clock enable (bit-12 of CLK_OUT_ENB.L register) must also be given to I2C controller, before any of the registers are written.

21.3.2 I2C Slave ADDR Registers

There are two slave ADDR registers that are used to configure the address of the internal slave-- I2C_SL_ADDR1 and I2C_SL_ADDR2. The bit [0] of I2C_SL_ADDR2 determines the address-size of the internal slave. When this bit is programmed to be zero, the 7-bit slave address is taken from I2C_SL_ADDR1 [6:0]. When this bit is programmed to be one, the two most significant bits of the 10-bit slave address are taken from I2C_SL_ADDR2[2:1] and the least significant bits are taken from I2C_SL_ADDR1[7:0].

21.3.3 I2C Command ADDR Registers

These registers, I2C_CMD_ADDR0 and I2C_CMD_ADDR1, contain the address of the slave with which a transaction is intended. I2C Master Controller can be programmed to transact with both 7-bit and 10-bit addressed slaves. The bit [0] of I2C_CNFG register determines the size of the slave address to be transacted. If a 7-bit or a 10-bit slave address is chosen, the respective address is taken from I2C_CMD_ADDR0 [9:0] and the Read/Write command is taken from the bits 6 & 7 of the I2C_CNFG Register. In a 2 Slave configuration, the slave 1 address (7-bit or 10-bit) is taken from I2C_CMD_ADDR0 [9:0] and Slave 2 address (7-bit or 10-bit) from I2C_CMD_ADDR1 [9:0].

21.3.4 I2C Data Registers

There are two data registers, I2C_CMD_DATA1and I2C_CMD_DATA2, each of 32 bit length, which are to be loaded with the data to be transmitted or received as an I^2C Master. When used as I2C slave, the controller uses a separate byte-wide register to store the data to be transmitted (I2C_SL_SEND) or data received (I2C_SL_RCVD).

21.3.5 I2C Configuration Register

This register is to be configured with the following data:

- Number of bytes to be transmitted or received
- Select for 7 bit or 10 bit slave address
- Program to send a Start-Byte
- Single or two slave operations
- Support of noack from an external Peripheral

The bit I2C_CNGF [9] is used to issue a “Begin-Transaction” command to the I2C Master Controller. This bit is reset by hardware and other bits of the register are masked for writes, when this bit is programmed to be one. Hence, the firmware should first configure all the other registers and the bits [8:0] of I2C_CNFG register before the bit I2C_CNFG [9] is programmed to one.

21.3.6 I2C Controller STATUS Register

This register (I2C_STATUS) gives the status of the I^2C Master operation. It includes the busy status of the I^2C Bus and the completion status of the command executed.

21.3.7 I2C Controller Slave Status Register

This register (I2C_SL_STATUS) contains the status bits as I^2C Slave, including Receive Interrupt, New Transaction Received and direction of transfer. Note that the I^2C Controller generates an interrupt at the completion of each data byte received. In response to this interrupt, firmware must read I2C_SL_STATUS register to confirm the cause of the interrupt and the direction of data transfer. If the interrupt is due to a data byte received (R/W=0), firmware must read the data register, I2C_SL_RCVD in order to clear the SL.IRQ bit in the status register. The I^2C Controller will delay the release of the ACK handshake on the I^2C bus until the SL.IRQ bit has been cleared by this firmware read operation.

I2C_SL_STATUS needs to be either polled or the I2C interrupt must be enabled for transfers to occur in slave mode. The following steps outline slave operation in both read and write transactions.

If Tegra 2 as a slave, is **receiving** data,

- I2C_SL_STATUS.SL_IRQ bit being set indicates a data byte has been received.
- Read to I2C_SL_STATUS clears SL_IRQ and returns the transfer direction in RNW field.
- Data byte is already present in I2C_SL_RCVD register. This can be read which causes the bus to be released.
- SL_IRQ is set again upon reception of next byte

If Tegra 2 as a slave, is **transmitting** data,

- I2C_SL_STATUS.SL_IRQ bit being set indicates that address has been received.
- Read to I2C_SL_STATUS clears SL_IRQ and returns the transfer direction in RNW field.
- Data byte needs to be written to I2C_SL_RCVD register. This write causes the bus to be released.
- SL_IRQ is set again upon transfer of the byte. Another byte can be written to continue the operation

21.3.8 Interrupt Generation

The I^2C controller generates interrupts upon the completion of transmission or reception of the specified number of bytes. A Master-interrupt is generated when the number of bytes of transaction, as programmed in the LENGTH field of the I2C_CNFG register, is complete.

A Slave-interrupt is generated at the transmission/reception of each byte of data by the internal slave. In addition, if the internal Slave is in transmitter mode (Slave receives a Read command), an interrupt is generated for the address and read-command reception also, as shown in the next two figures below.. Slave interrupts are to be cleared by the firmware as mentioned above.

Figure 40 Interrupt Generation: Master Transmits to Internal Slave

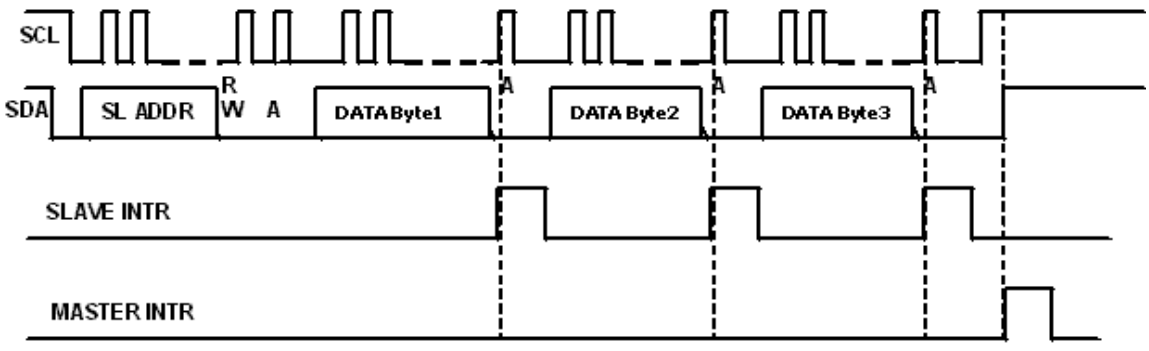
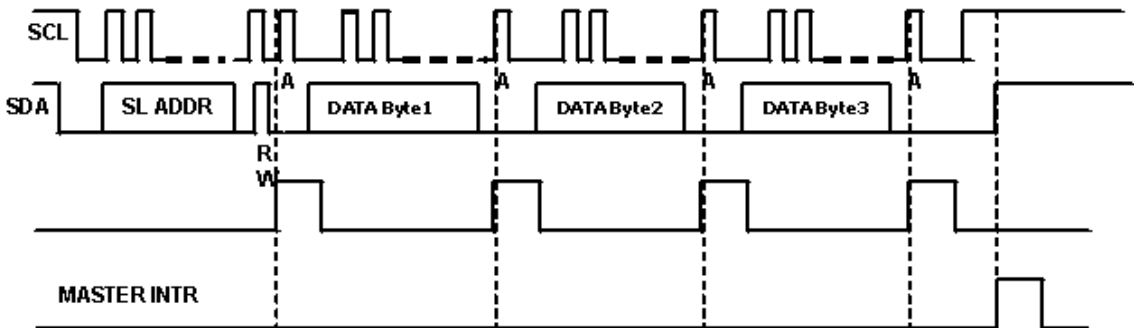


Figure 41 Interrupt Generation: Master Reads from Internal



21.4 Programming Guidelines for Packet Based Interface

21.4.1 Packet Based Interface Registers

The following registers need to be programmed in case of packet based interface:

- I2C TX Packet FIFO Register
- I2C RX FIFO
- PACKET_TRANSFER_STATUS
- FIFO CONTROL
- FIFO STATUS
- INTERRUPT_MASK_REGISTER
- INTERRUPT_STATUS_REGISTER
- PACKET_MODE_EN , DEBOUNCE_CNT and NEW_MASTER_FSM field of I2C Controller Configuration Register

All other registers/fields have no meaning in packet mode and should be used only normal mode.

21.4.2 Programming Packet Header and Payload

SW should program the I2C_TX_PACKET_FIFO register. It writes packet header followed by payload. Header size is variable, could vary from 3 to 5 words. For I2C, it is 3 words. The first two words of the header contain generic header. The third word of the Packet contains I2C transaction specific information. Payload contains the actual data to be written to the slave. In case of read operation, payload is nil, and hence, packet contains only header.

21.4.3 Reading from Rx_fifo

The data received on the bus is pushed into rx_fifo. In PIO mode A, a request interrupt is generated when the attention level is reached and SW reads rx_fifo register to get the data. In DMA mode, a DMA trigger is asserted after reaching the attention level.

21.4.4 Error Handling

In case of an error due to nack or loss of arbitration, the following steps describe what happens when an error occurs:

- A stop condition is put on the bus and interrupt is generated
- The TX(packet)_FIFO will be reset
- Status register is updated with the current error packet id, and the byte number at which the error occurred.
- SW should reset the controller

- In case of DMA transfer, DMA needs to be restarted

In packet mode special care should be taken to handle error recovery. If an error occurs during the transfer of a packet then the entire packet needs to be resent since there is no accurate way of knowing how many bytes made it to the I2C slave.

Repeat start bit in packet mode can complicate the situation further since when back to back packets are sent to a slave with repeat start bit set, and an error occurs then we can't reliably know during which packet transfer the error occurred. Therefore the entire stream of packets that were sent back to back using repeat start need to be sent again.

On the receive case it is a bit simpler since if an error occurs, and bytes for a packet have already been received then that packet can be deemed complete. Only incomplete transfers need to be retried.

21.4.5 Programming repeat_start/stop

This field indicates whether to put a stop condition after the current transaction. By default, this bit is zero and a stop condition is put on the bus. If this bit is set, a repeat start is put on the bus before proceeding with the next packet. Repeat_start/stop bit can be used for combining read operations and write operations within a single transaction with repeat start, or to do transfers beyond the 4Kbyte limit. This bit is present in the protocol specific header.

21.4.6 FIFO Control

In DMA mode, TX_FIFO_TRIG indicates the number of words that need to be empty in TX FIFO for the DMA trigger to be asserted. RX_FIFO_TRIG indicates the number of words that need to be full in RX FIFO for the DMA trigger to be asserted.

In PIO mode, TFIFO_DATA_REQ in INTERRUPT_STATUS_REGISTER will be set, if the TX FIFO empty count is more than the value programmed in TX_FIFO_TRIG. Similarly RFIFO_DATA_REQ in INTERRUPT_STATUS_REGISTER will be set if RX FIFO full count is more than the value programmed in RX_FIFO_TRIG.CPU will be interrupted if status bits are set and their corresponding INT_EN is set in the INTERRUPT_MASK_REGISTER. The depth of TX and RX FIFOs is 8.

21.4.7 FIFO Status

This register indicates the number entries that are empty in the TX FIFO (TX_FIFO_EMPTY_CNT) and the number of entries that are full in the RX FIFO(RX_FIFO_FULL_CNT).

21.4.8 Example Transfer

A programming example for 1 byte read followed by 5 byte write

1. Program the PACKET_MODE_EN field to 1
2. Write the generic packet header
3. Write payload size = 0x0(which means 1 byte)
4. Write the i2c specific header read/write = 1 and repeat_start/stop =1
5. Write the generic packet header
6. Write payload size = 0x4(which means 5 bytes)
7. Write the i2c specific header read/write = 0 and repeat_start/stop
8. Write the 12 bytes(word aligned)

21.5 I2C Registers

21.5.1 I2C_I2C_CNFG_0

IC Controller Configuration Register (Master) I2C_CNFG register is used to configure, the number of bytes to be transmitted or received, the slave device type either a 7-bit device or a 10-bit device, enable mode to send Start-Byte or not, to select either a single slave transaction or two slave transaction, Enable mode to handle devices that do not generate ACK.

Offset: 000h | Read/Write: R/W | Reset: 0b0000000000000000

Bit	Reset	Description
14:12	0x0	DE-BOUNCE_CNT: De-bounce period for sda and scl lines 0 = No De-bounce 1 = 2T 2 = 4T 3 = 6T etc where T is the period of the fix PLL clk source coming to i2c.Maximum De-bounce period programmable is 14T. A De-bounce period of >50ns is desirable
11	0x0	NEW_MASTER_FSM: Write 1 to enable new master fsm 0 = old fsm 0 = DISABLE 1 = ENABLE
10	0x0	PACKET_MODE_EN: Write 1 to initiate transfer in packet mode. 0 = NOP 1 = GO
9	0x0	SEND: Writing a 1 causes the master to initiate the transaction in normal mode. Values of other bits are not affected when this bit is 1, Cleared by hardware. Other bits of the register are masked for writes when this bit is programmed to one. Hence, firmware should first configure all other registers and bits [8:0] of I2C_CNFG register before the bit I2C_CNFG[9] is programmed to Zero. 0 = NOP 1 = GO
8	0x0	NOACK: Enable mode to handle devices that do not generate ACK. 1 - don't look for an ack at the end of the Enable 0 = DISABLE 1 = ENABLE
7	0x0	CMD2: Read/Write Command for Slave 2: 1 - Read Transaction; 0 - write Transaction. For a 7-bit slave address, this bit must match with the LSB of address byte for slave 2. Valid only when bit-4 of this register is set 0 = DISABLE 1 = ENABLE
6	0x0	CMD1: Read/Write Command for Slave 1: 1 - Read Transaction; 0 - write Transaction. Command for Slave 1: For a 7-bit slave address this bit must match with the LSB of address byte for slave1. 0 = DISABLE 1 = ENABLE
5	0x0	START: 1 = Yes, a Start byte needs to be sent. 0 = DISABLE 1 = ENABLE
4	0x0	SLV2: 1 - Enables a two slave transaction; 0 = No command for Slave 2 present. 0 = DISABLE 1 = ENABLE
3:1	0x0	LENGTH: The Number of bytes to be transmitted per transaction 000= 1byte ... 111 = 8bytes; In a two slave transaction number of bytes should be programmed less than 011.
0	0x0	A_MOD: Address mode defines whether a 7-bit or a 10-bit slave address is programmed. 1 = 10-bit device address 0 = 7-bit device address 0 = SEVEN_BIT_DEVICE_ADDRESS 1 = TEN_BIT_DEVICE_ADDRESS

21.5.2 I2C_I2C_CMD_ADDR0_0

I2C_CMD_ADDR0 is programmed the 7 Bit or 10 Bit address of slave 1 with which the transaction is intended.

I2C Slave-1 Address

Offset: 004h | Read/Write: R/W | Reset: 0b0000000000

Bit	Reset	Description
9:0	0x0	ADDR0: In case of 7-Bit mode address is written in the I2C_CMD_ADDR0[7:1] and I2C_CMD_ADDR0[0] indicates the read/write transaction.I2C_CMD_ADDR0[0] bit must match with the I2C_CNFG[6]. In case of 10-Bit mode address is written in I2C_CMD_ADDR0[9:0] and I2C_CNFG[6] indicates the read/write transaction.

21.5.3 I2C_I2C_CMD_ADDR1_0

I2C_CMD_ADDR1 is programmed the 7 Bit or 10 Bit address of slave 2 with which the transaction is intended.

I2C Slave-2 Address

Offset: 008h | Read/Write: R/W | Reset: 0b0000000000

Bit	Reset	Description
9:0	0x0	ADDR1: In case of 7-Bit mode address is written in the I2C_CMD_ADDR0[7:1] and I2C_CMD_ADDR0[0] indicates the read/write transaction.I2C_CMD_ADDR0[0] bit must match with the I2C_CNFG[7]. In case of 10-Bit mode address is written in I2C_CMD_ADDR0[9:0] and I2C_CNFG[7] indicates the read/write transaction.

21.5.4 I2C_I2C_CMD_DATA1_0

The four Least Significant Bytes of Data to be transmitted is loaded into the register when I2c Master is in Write Mode.

The four Least Significant Bytes of Data are Read through this register when I2C Master is in Read mode.

IC Controller Data 1: Transmit/Receive

Offset: 00ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:24	0x0	DATA4: Fourth data byte to be sent/received
23:16	0x0	DATA3: Third data byte to be sent/received
15:8	0x0	DATA2: Second data byte to be sent/received
7:0	0x0	DATA1: This register contains the first data byte to be sent/received.

21.5.5 I2C_I2C_CMD_DATA2_0

The four Most Significant Bytes of Data to be transmitted is loaded into the register when I2C Master is in Write Mode.

The four Most Significant Bytes of Data are Read through this register when I2C Master is in Read mode.

IC Controller Data 2: Transmit/Receive

Offset: 010h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:24	0x0	DATA8: Eighth data byte to be sent/received
23:16	0x0	DATA7: Seventh data byte to be sent/received
15:8	0x0	DATA6: Sixth data byte to be sent/received
7:0	0x0	DATA5: This register contains the Fifth data byte to be sent/received.

21.5.6 I2C_I2C_STATUS_0

I2C_STATUS gives the status of I2C master operation.

IC Controller Status (Master)

Offset: 01ch | Read/Write: RO | Reset: 0bxxxxxxxxx

Bit	Reset	Description
8	X	BUSY: 1 = Busy. 0 = NOT_BUSY 1 = BUSY

Bit	Reset	Description
7:4	X	CMD2_STAT: Transaction for Slave2 for x byte failed. x is 'h0 to 'ha. all others invalid 0 = SL2_XFER_SUCCESSFUL 1 = SL2_NOACK_FOR_BYTE1 2 = SL2_NOACK_FOR_BYTE2 3 = SL2_NOACK_FOR_BYTE3 4 = SL2_NOACK_FOR_BYTE4 5 = SL2_NOACK_FOR_BYTE5 6 = SL2_NOACK_FOR_BYTE6 7 = SL2_NOACK_FOR_BYTE7 8 = SL2_NOACK_FOR_BYTE8 9 = SL2_NOACK_FOR_BYTE9 10 = SL2_NOACK_FOR_BYTE10
3:0	X	CMD1_STAT: Transaction for Slave1 for x byte failed. x is 'h0 to 'ha. all others invalid 0 = SL1_XFER_SUCCESSFUL 1 = SL1_NOACK_FOR_BYTE1 2 = SL1_NOACK_FOR_BYTE2 3 = SL1_NOACK_FOR_BYTE3 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

21.5.7 I2C_I2C_SL_CNFG_0

I2C_SL_CNFG register is used to configure,

- Enable mode of slave Ack
- Enable mode of slave response to general call address

The register should be programmed when I2C controller is configured as slave.

IC Controller Configuration (Slave)

Offset: 020h | Read/Write: R/W | Reset: 0b000

Bit	Reset	Description
2	0x0	NEWSL: New Slave 1 - use new slave 0 = DISABLE 1 = ENABLE
1	0x0	NACK: Disable Slave Ack. 1 - slave will not ack reception of address or data byte. 0 = DISABLE 1 = ENABLE
0	0x0	RESP: Slave response to general call address (zero address) 1 - Enable. 0 = DISABLE 1 = ENABLE

21.5.8 I2C_I2C_SL_RCVD_0

IC Controller Slave Receive/Transmit Data (Slave)

Offset: 024h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7:0	0x0	SL_DATA: slave Received data

21.5.9 I2C_I2C_SL_STATUS_0

IC Controller Slave Status (Slave)

Offset: 028h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxx

Bit	Reset	Description
14:8	X	HW_MSTR_ADR: HW master addr received via general call addressing. This field is meaningful only if HW_MSTR_INT is set.
7	X	HW_MSTR_INT: 1 = Interrupt has been generated by slave Hardware Master Address is received after General Call Address. 1 = Received HW Master Address 0 = No event.
6	X	REPROG_SL: 1 = Interrupt has been generated by slave after General Call Address is 0x04. 1 = Reprogram slave address. 0 = No action.
5	X	RST_SL: 1 = Interrupt has been generated by slave after General Call Address is 0x06. 1 = Reset and reprogram slave address. 0 = No action.
4	X	END_TRANS: 1 = Interrupt has been generated by slave Transaction completed as indicated by stop/repeat start condition. 1 = Transaction completed. 0 = No transaction occurred or transaction in progress.
3	X	SL_IRQ: 1 = Interrupt has been generated by slave 0 = No interrupt generated 0 = UNSET 1 = SET
2	X	RCVD: New Transaction Received status 1 = Transaction occurred. 0 = No transaction occurred 0 = NO_TRANSACTION_OCCURED 1 = TRANSACTION_OCCURED
1	X	RNW: Slave Transaction status 0 = Write 1=Read 0 = WRITE 1 = READ
0	X	ZA: Zero Address Status 1 = Yes, slave responded 0 = No, slave did not respond 0 = NO_SLAVE_RESPONSE 1 = SLAVE_RESPONSE

21.5.10 I2C_I2C_SL_ADDR1_0

IC Controller Slave Address 1 Register (Slave)

Offset: 02ch | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7:0	0x0	SL_ADDR0: For a 10-bit slave address, this field is the least significant 8 bits.

21.5.11 I2C_I2C_SL_ADDR2_0

IC Controller Slave Address 2 Register (Slave)

Offset: 030h | Read/Write: R/W | Reset: 0b000

Bit	Reset	Description
2:1	0x0	SL_ADDR_HI: In 7 bit address mode these bits are don't care; In 10 bit address mode they represent the 2 MSB of the address.
0	0x0	VLD: 0 = 7-bit addressing. 1 - 10 bit addressing. 0 = SEVEN_BIT_ADDR_MODE 1 = TEN_BIT_ADDR_MODE

21.5.12 I2C_I2C_SL_DELAY_COUNT_0

IC Slave Controller Delay Count

Offset: 03ch | Read/Write: R/W | Reset: 0b0000000000011110

Bit	Reset	Description
15:0	0x1e	SL_DELAY_COUNT: The value determines the timing between an address cycle and a subsequent data cycle or two consecutive data cycles on the bus. The I2C_SL_DELAY_COUNT is valid only when internal slave is accessed. I2C_SL_DELAY_COUNT has to be programmed such that TIMING = T * DLY where T is period of clock source selected for I2c; and DLY is I2C_SL_DELAY_COUNT; TIMING is the desired timing, A value of >= 1250 ns is advisable.

Imp. Note: Please program the field PACKET_MODE_EN of I2C_CNFG register while working in packet mode.

The set of registers below describe the interface for packet mode only. The registers present in the ari2cm.spec describe the interface for normal mode. With the packet mode interface changes, normal mode registers and the operation are not affected. The transition from normal mode to packet mode is suggested because packet mode allows:

1. There is no restriction on the no. of bytes before and the after the repeated start
2. The number of bytes that can be transferred with a single cmd is not limited to 8. Though a packet can contain 4k bytes, because any number of packets can be pushed into the fifo, there is no limit on the no. of bytes that can be transferred.
3. The transactions to different slaves can be chained together with repeat start and there is no limit on the no. of slaves it can address.

21.5.13 I2C_I2C_TX_PACKET_FIFO_0

A packet contains header and payload. Header size is variable and could vary from 2 to 5 words. For I2C it is 3 words. The first two words of the header contain generic information. The third word contains I2C transaction specific information. Payload contains actual data to be written to the slave. In case of read operation, payload is nil, hence the packet contains header only.

Offset: 050h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	TX_PACKET: SW writes packets into this register A packet may contain generic

21.5.14 I2C_I2C_RX_FIFO_0

Header or I2C Specific Header or Data

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

Bit	Reset	Description
31:0	X	RD_DATA: SW Reads data from this register, causes pop

21.5.15 I2C_PACKET_TRANSFER_STATUS_0

Offset: 058h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
24	X	TRANSFER_COMPLETE: The packet transfer for which last packet is set has been completed 0 = UNSET 1 = SET
23:16	X	TRANSFER_PKT_ID: The current packet id for which the transaction is happening on the

		bus
15:4	X	TRANSFER_BYTENUM: The number of bytes transferred in the current packet
3	X	NOACK_FOR_ADDR: No ack received for the addr byte 0 = UNSET 1 = SET
2	X	NOACK_FOR_DATA: No ack received for the data byte 0 = UNSET 1 = SET
1	X	ARB_LOST: Arbitration lost for the current byte 0 = UNSET 1 = SET
0	X	CONTROLLER_BUSY: 1 = Controller is busy 0 = UNSET 1 = SET

21.5.16 I2C_FIFO_CONTROL_0

Offset: 05ch | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7:5	0x0	TX_FIFO_TRIG: Transmit FIFO trigger level 000 = 1 word, DMA trigger is asserted when at least one word empty in the FIFO 010 = 2 word, DMA trigger is asserted when at least 2 words empty in the FIFO
4:2	0x0	RX_FIFO_TRIG: Receive FIFO trigger level 000 = 1 word DMA trigger is asserted when at least one word full in the FIFO 010 = 2 word DMA trigger is asserted when at least 2 word full in the FIFO
1	0x0	TX_FIFO_FLUSH: 1= flush the TX FIFO, cleared after FIFO is flushed 0 = UNSET 1 = SET
0	0x0	RX_FIFO_FLUSH: 1= flush the RX FIFO, Cleared after FIFO is flushed 0 = UNSET 1 = SET

21.5.17 I2C_FIFO_STATUS_0

Offset: 060h | Read/Write: RO | Reset: 0bxxxxxxxx

Bit	Reset	Description
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 fifo0000 = rx_fifo empty0001 = 1 slot full0010 = 2 slots full

21.5.18 I2C_INTERRUPT_MASK_REGISTER_0

Offset: 064h | Read/Write: R/W | Reset: 0b0000000

Bit	Reset	Description
6	0x0	ALL_PACKETS_XFER_COMPLETE_INT_EN: 0 = DISABLE 1 = ENABLE
5	0x0	TFIFO_OVF_INT_EN: 0 = DISABLE 1 = ENABLE
4	0x0	RFIFO_UNF_INT_EN: 0 = DISABLE 1 = ENABLE
3	0x0	NOACK_INT_EN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
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

21.5.19 I2C_INTERRUPT_STATUS_REGISTER_0

This register indicates the status bit for which the interrupt is set. If set, Write 1 to clear it However TFIFO_DATA_REQ,RFIFO_DATA_REQ fields depend on the fifo trigger levels and cannot be cleared.

Offset: 068h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	PACKET_XFER_COMPLETE: A packet has been transferred successfully. TRANSFER_PKT_ID filed can be used to know the current byte under transfer. This bit can be masked by the IE field in the i2c specific header 0 = UNSET 1 = SET
6	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
1	0x0	TFIFO_DATA_REQ: Tx fifo data req 0 = UNSET 1 = SET
0	0x0	RFIFO_DATA_REQ: rx fifo data req 0 = UNSET 1 = SET

21.5.20 I2C_I2C_CLK_DIVISOR_REGISTER_0

The divisor value (N) must be programmed so that desired scl freq = Clk source freq/12*N. Typically clk source frequency is fixed at 72 MHz.

Offset: 06ch | Read/Write: R/W | Reset: 0b0000000000000000

Bit	Reset	Description
15:0	0x0	I2C_CLK_DIVISOR_HSMODE: N= divide by n+1

22.0 UART AND VFIR CONTROLLER

There are five Universal Asynchronous Receiver/Transmitters (UARTs) built into the Tegra® 2 Processor. These UARTs support both 16450 and 16550 compatible modes. All UARTs are identical, though their pin connections will be different. One of the UARTs also contains VFIR functionality.

All UARTs provides serial data synchronization and data conversion (parallel-to-serial and serial-to-parallel) for both receiver and transmitter sections. Synchronization for serial data stream is accomplished by adding start and stop bits to the transmit data to form a data character. Data integrity is accomplished by attaching a parity bit to the data character. The parity bit can be checked by the receiver for any transmission bit errors.

The COM-HOST interface is fully programmable through an 8-bit CPU interface. The registers are 32-bit word aligned. The interface supports word lengths from five to eight bits, an optional parity bit and one or two stop bits. If enabled, parity can be odd, even, or forced to a defined state. Interrupts can be generated from any of 10 sources.

The UARTs support both 16450 and 16550 compatible modes. The default mode is 16450. This mode provides independent 16-byte FIFOs for transmit and receive and is selected by means of the FIFO control register. It also includes a 16-bit programmable baud rate generator and an 8-bit scratch register, eight modem control lines, and two DMA handshake lines that are used to indicate when the FIFOs are ready to transfer data to the CPU. The UARTs support a device clock of up to 72 MHz. The maximum baud rate is 16 clock cycles per symbol, and is therefore 4.5 Megabits per second.

Note: Mention of UART A/B/C/D/E in this document equate to UART 1/2/3/4/5 respectively.

The Tegra 2 VFIR controller implements the control and data sections of the IrDA Physical layer version 1.4, handling handshaking and basic networking between devices, and interfaces to an external Infrared Transceiver.

This controller implements three distinct speed ranges and protocols: 2400 to 115,200 bits per second (Serial IR or SIR), 4 Megabits per second (called Fast Infrared or FIR), and 16 Megabits per second (called Very Fast Infrared or VFIR). Initial communication starts at 9600 bits per second, and negotiations proceed either faster or slower as the link strength allows. Each of the three speed ranges has different communication protocols and encoding schemes.

Table 65 Protocols Supported by the IrDA

Signaling Rate (BPS)	IrDA Name	Frame Wrapper and Encoding	IrDA Status	Tegra 2 Series Support
9.6 K	SIR	Async RZI (3/16)	Mandatory	Yes
2400 - 115.2K	SIR	Async RZI (3/16)	Recommended	Yes
576.0 K	MIR	Sync HDLC RZI (4/16)	Optional	No
1.152 M	MIR	Sync HDLC RZI (4/16)	Optional	No
4.0 M	FIR	Sync 4PPM	Optional	Yes
16.0 M	VFIR	Sync HHH(1,13)	Optional	Yes

22.1 Hardware Features

- Synchronization for the serial data stream with start and stop bits to transmit data to and from a data character
- Data integrity by attaching parity bit to the data character
- The COM-HOST interface is fully programmable through an 8-bit CPU interface
- Support for word lengths from five to eight bits, an optional parity bit and one or two stop bits

- The UART supports both 16450- and 16550-compatible modes. Default mode is 16450
- DMA capable for both TX and RX
- 8 bit x 16 deep FIFO
- Auto sense baud detection

The features supported by the VFIR controller are:

- Supports IrDA ver1.0 SIR protocol with maximum baud rate to 115.2 Kilobits per second
- Supports IrDA ver1.3 FIR protocol (4 Megabits per second)
- Supports IrDA ver1.4 VFIR protocol (16 Megabits per second)
- Programmable polarities on all the interface pins
- DMA capable for both TX and RX
- 32 bit x 16 deep FIFO
- Diagnostics for loopback and error insertion
- Interface control, such as transceiver signal strength, must be performed through GPIOs
- Maximum frequency of device clock is 72 MHz

22.2 Hardware Signaling

All of Tegra's UARTs are implemented by a HW block that supports modem control signals such as DTR, DSR, DCD. Only UART 1 allows those signals to be mux'd externally.

For UART A, when the pinmux is configured for 4- or 2-pin UART operation, the DSR and DCD input signals are tied low (active).

For UART B, the DTR output is routed back into the DSR and DCD inputs in HW. This means that if SW changes DTR state, the UART may raise an interrupt due to DSR/DCD changing state.

For UARTs C, D, and E, the DSR and DCD inputs are tied to an inactive value.

Table 66 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.

22.3 Functionality

The controller can operate in three different encodings and each is implemented separately. In SIR mode, a pulse encoder/decoder is inserted in the transmit/receive path of a standard UART. FIR and VFIR modes have their own control blocks. The outputs of the three blocks are multiplexed based on the current operating mode (SIR, FIR, or VFIR), before they leave the Tegra 2 Series processor. The input signals are de-multiplexed in a similar fashion.

Note:	The operating mode is selected with the Mode [2:0] bits of the VFIR.CTL register

Figure 42 UART Block Diagram with SIR Decoder and Encoder

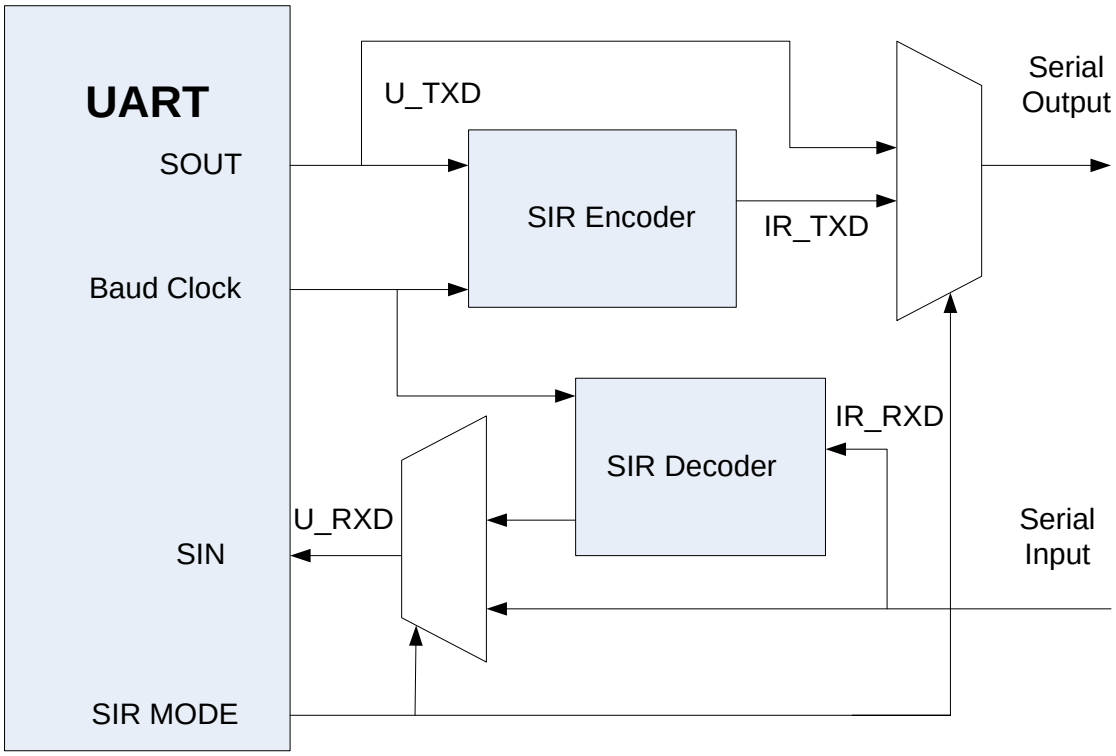
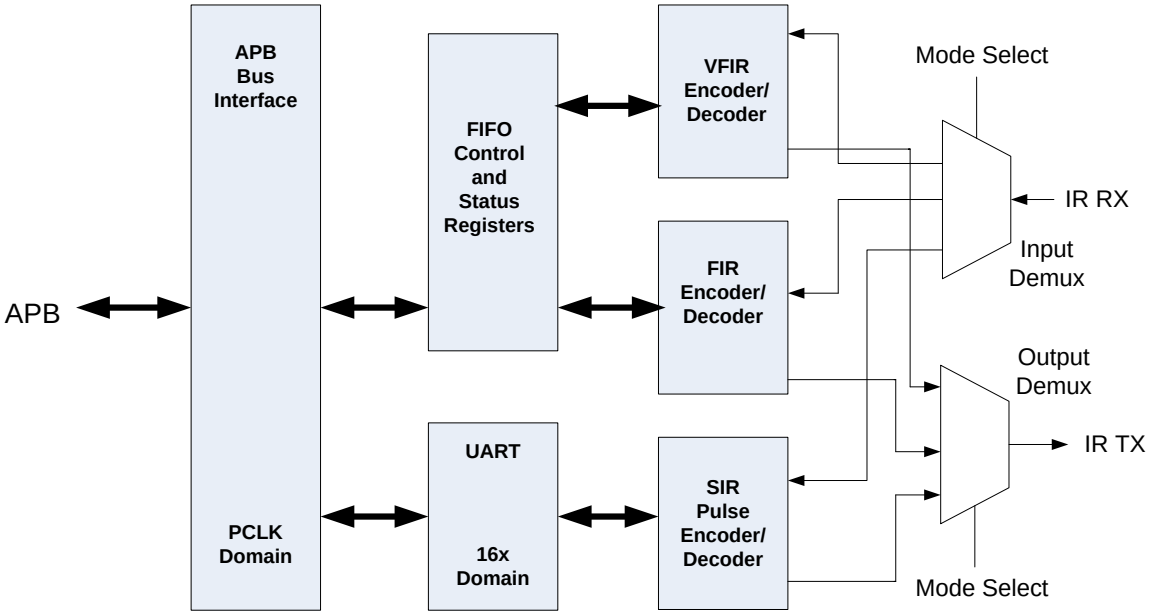


Figure 43 VFIR Controller Top-level Block Diagram



22.4 UART Programming Guidelines

22.4.1 16450 Mode Programming

1. Program pin mux settings to select UART
2. Enable UART clocks
3. For enabling internal loop-back, program MCR[4] to 1
4. Program FCR[0] to 0
5. Enable interrupts in IER register as needed
6. Write data into THR
7. Wait for THR interrupt if Interrupt is enabled or poll for LSR[5]
8. During receive, wait for RBR interrupt or poll for LSR[0]
9. Read the UART.LSR register to clear interrupts

22.4.2 16550 Mode Programming

1. Program pin mux settings to select UART
2. Enable UART clocks
3. For enabling internal loop-back, program MCR[4] to 1
4. Program FCR[0] to 1
5. Program trigger levels as required
6. Enable interrupts in IER register as needed
7. Write data into THR
8. Wait for THR interrupt if Interrupt is enabled or poll for LSR[5]
9. During receive wait for RBR interrupt or poll for LSR[0]
10. Read the UART.LSR register to clear interrupts
11. Apb-dma requestor number for UARTA, B and C are 8, 9 and 10 respectively

22.4.3 Transmitter and Receiver Holding Registers

The serial transmitter section consists of a Transmit Hold Register (THR) and Transmit Shift Register (TSR). The status of transmit hold register is provided in the Line Status Register (LSR). Writing to this register (THR) transfers the contents of data bus (D[7:0]) to the transmit holding register whenever the transmitter holding register or transmitter shift register is empty. The transmit holding register empty flag is set to 1 when the transmitter is empty or data is transferred to the transmit shift register. Note that a write operation is performed when the transmit holding register empty flag is set.

The serial receiver section also contains an 8-bit Receiver Holding/Buffer Register (RBR). Receive data is removed from the UART and received by the processor by reading the RBR. The receiver contains a mechanism for preventing false starts as follows: On the falling edge of the start bit, the receiver internal counter starts to count 7.5 clocks (16x clock), which is the center of the start bit. The start bit is valid if the RX is still low at the mid-bit sample of the start bit. Verifying the start bit prevents the receiver from assembling a false data character due to a low going noise spike on the RX input. Receiver status codes are posted in the line status register.

22.4.4 FIFO Interrupt Mode Operation

When the receive FIFO (FCR bit 0 = 1) and receive interrupts (IER bit 0 = 1) are enabled, a receiver interrupt occurs as follows:

- The receive data available interrupts are issued to the CPU when the FIFO has reached its programmed trigger level; it is cleared as soon as the FIFO drops below its programmed trigger level.
- The ISR receive data available indication also occurs when the FIFO trigger level is reached, and like the interrupt, it is cleared when the FIFO drops below the trigger level.
- The data ready bit (LSR bit 0) is set as soon as a character is transferred from the shift register to the receiver FIFO. It is reset when the FIFO is empty.

22.4.5 FIFO Polled Mode Operation

When FCR bit 0 = 1; clearing IER bits [3:0] to zero puts the UART in the FIFO polled mode of operation. Since the receiver and transmitter are controlled separately, either one or both can be in the polled mode operation by using the line status register as follows:

- LSR bit 0 is set as long as there is one byte in the receive FIFO
- LST bits [4:1] specifies which error(s) have occurred
- LSR bit 5 indicates when the transmit FIFO is empty
- LSR bit 6 indicates when both transmit FIFO and transmit shift registers are empty
- LSR bit 7 indicates when there are any errors in the receive FIFO

The UART requires a two-step FIFO enable operation in order to enable receive trigger levels.

22.4.6 Programmable Baud Rate Generator

The UART contains a programmable baud rate generator that is capable of taking the UART clock input and dividing it by any divisor from 1 to 2^16-1. The UART clocks are defined in section 6.1 on Clocking. The output frequency of baudout is equal to 16X the transmission baud rate (Baudout=16 X baud rate). Customized baud rates are achieved by selecting proper divisor values for the MSB and LSB bits of the baud rate generator.

22.4.7 Enable Register (IER)

The is an Interrupt Enable Register for each UART (UART A Interrupt Enable and UART B Interrupt Enable). The Interrupt Enable Register(s) masks the incoming interrupts from the receiver ready, transmitter empty, line status and modem status registers to the INT output pin.

22.4.8 Interrupt Identification Register (IIR)

The UART provides four level prioritized interrupt conditions to minimize software overhead during data character transfers. The Interrupt Identification Register (IIR) provides the source of the interrupt in a prioritized manner.

During the read cycle the 16550 provides the highest interrupt level to be serviced by the CPU. No other interrupts are acknowledged until the particular interrupt is serviced. The prioritized interrupt levels are shown in the following table. The Receive Data Time-out mode is enabled when the UART is operating in the FIFO mode.

Receive time-out does not occur if receive FIFO is empty. The time-out counter is reset at the center of each stop bit received or each time the receive holding register is read. The actual time out value is:

- T (Time out length in bits) = 4 X P (Programmed work length) + 12

To convert the time-out value to a character value, divide this number to its complete word length + parity (if used) + number of stop bits and start bit.

Example

If you program the word length = 7, no parity, and one stop bit, the time-out is:

- T = 4X7 (programmed word length) + 12 = 40 bits.
- Character time = 40/9
- [(Programmed word length = 7) + (stop bit = 1) + (start bit = 1)] =4.4 characters

Table 67 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)

22.4.9 FIFO Control Register (FCR) Modes

The FIFO control register is used to enable the FIFOs, clear the FIFOs, set the receiver FIFO trigger level, and select the type of DMA signaling. The operation of the FCR in the four DMA modes is given below.

22.4.9.1 Transmit operation in DMA mode 0 or mode 1

When the UART is in the 16450 mode (FCR bit 0 = 0) or in the FIFO mode (FCR bit 0 = 1, FCR bit 3 = 0) and when there are no characters in the transmit FIFO or transmit holding register, the TXRDY* pin goes low. Once active, the TXRDY* pin goes high (inactive) after the first character is loaded into the transmit holding register.

22.4.9.2 Receive operation in DMA mode 0

When UART is in 16450 mode (FCR bit 0 = 0) or in the FIFO mode (FCR bit 0 = 1,FCR bit 3 = 0) and there is at least 1 character in the receive FIFO, the RXRDY* pin goes low. Once active, the RXRDY* pin goes high (inactive) when there are no more characters in the receiver.

22.4.9.3 Receive operation in DMA mode 1

When UART is in the FIFO mode (FCR bit 0 = 1, FCR bit 3 = 1) and the trigger level or the timeout has been reached, the RXRDY* pin goes low. Once it is activated it goes high (inactive) when there are no more characters in the FIFO.

22.4.10 Line Control Register (LCR)

The Line Control Register is used to specify the asynchronous data communication format. The word length, stop bits, and parity can be selected by writing appropriate bits in this register.

22.4.11 Modem Control Register (MCR)

This register controls the interface with the modem or a peripheral device (RS232).

Loopback Mode

If MCR[4] = 1, the loopback mode is enabled and the following occurs:

The transmitter output (TX) is set high (Mark condition), and the receiver input (RX), CTS, DSR, CD, and RI are disabled. Internally the transmitter output is connected to the receiver input and DTR, RTS, OP1 and OP2 are connected to modem control inputs.

In this mode, the receiver and transmitter interrupts are fully operational, but the interrupt sources are now the lower four bits of the modem control register instead of the four modem control inputs. The interrupts are controlled by the IER register.

Table 68 Modem Control Register (MCR)

Bit	Name	Description
7:6	N/A	Not used. Set to 0 internally.
5	CTS.EN	1 = Enable Hardware Flow Control using CTS 0 = Disable Hardware Flow Control
4	LOOP	0 = Normal operating mode. 1 = Enable local loop-back mode (diagnostic).
3	OUT2	nOUT2 polarity
2	OUT1	nOUT1 polarity
1	RTS	1 = Force RTS (Request to Send) low 0 = Force RTS to high
0	DTR	1 = Force DTR (Data Terminal Ready) to low 1 = Force to high

22.4.12 Line Status Register (LSR)

The Line Status Register provides the status of data transfer to the CPU.

22.4.13 Modem Status Register (MSR)

The modem status register provides the current state of the control lines from the modem or peripheral to the CPU. Four bits of this register are used to indicate the changed information. These bits are set to 1 whenever a control input from the modem changes state. They are set to 0 whenever the CPU reads the register. The following table shows the bit format for the MSR register.

22.4.14 Scratchpad Register (SR)

Eight bits of information can be stored in this register. Information in this register does not affect the operation of the device in any way.

22.4.15 Autobaud Sense Register (ASR)

The UART has the ability to automatically determine the correct baud divisor by using the Autobaud Sense Register (UART offsets 0x3C). The most significant bit of this register is the valid flag. A write to this register will clear the valid flag and enable the autobaud process. When the first RX edge occurs, a counter running at 24 MHz starts counting and when another rx_edge occurs, the "complete" flag is set, the value frozen and the autobaud sense value register updated with the count value. The low 20 bits of the ASR gives the number of clocks within a single bit. Because the UART uses 16x over sampling, the resulting value needs to be adjusted by shifting right 4 bits, then loading the resulting count in the divisor latch of the UART. (In the code snippet below, the lower 4 bits are rounded to give slightly greater accuracy.)

Because the speed determination is made by measuring the start bit, special characters must be sent by the transmitting UART to guarantee that the next character after the start bit is a 1. Since bit 0 (rightmost bit) is sent first, the ASCII Carriage Return character (CR) is sufficient to enable proper speed sense.

The following code snippet returns the values for DLH and DLL in r9, r8:

```
       MOV     r0, #1  ; dummy write data
       STRB    r0, [r3, #U_ASR]; Start autobaud sense (r3 as uart_base)
; now poll the autobaud sense register MSB until Valid is truewait4valid
```

```
        LDR     r2, [r3, #U ASR]; Read ASR
        TST     r2, #0x80000000 ; the Valid bit (active high)
        BEQ     wait4valid
; autobaud sense check complete...
; r2 as number of 1x clocks in one bit time
; representing the number of 24MHz clocks in the start bit
; Since this represents 16 of the baud (16x) clocks, we
; will be dividing by 16 to get baud divisor, but first
; round to nearest by adding 8 before the divide:
        ADD     r4, r2, #8 ; add 1/2 resolution
        MOV     r6, r4, LSR #4 ; divide by 16... R6 will have total divisor
        AND     r8, r6, #0xFF  ; copy DLL to r8
        MOV     r9, r6, LSR #8 ; copy DLM to r9
        AND     r9, r9, #0xff ; mask upper bytes
```

22.4.16 Baud Rate Generator

The following table given below is a divisor table for the baud rate, assuming the oscillator is 24.000 MHz.

Table 69 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

22.4.17 Power-up Defaults

The following defines the Power-Up defaults:

- IER = 0
- ISR = 1
- LCR = 0
- MCR = 0
- LSR = 60 HEX
- MSR = BITS 0-3 = 0, BITS 4-7 = inputs
- FCR = 0
- TX = High
- OP1 = High
- OP2 = High
- RTS = High
- DTR = High

- RXRDY = High
- TXRDY = Low
- INT = Low

22.5 VFIR Programming Guidelines

22.5.1 IRDA (FIR/VFIR) and UART B

The IRDA module includes a fully functional 16550 compatible UART for implementing the SIR protocol. The UART pins will be selected automatically when the VFIR.CTL Mode bits are set to UART or SIR modes. Set the UART IRDA.CSR register to SIR mode to convert the UART signals to SIR signals.

If the IRDA.CTL Mode bits are set the FIR or VFIR modes, the UART pins will be disabled, and the FIR/VFIR pins will be used.

22.5.2 APB Bus Interface to Control/Status/FIFO

The Control and Status registers, and the FIFO reside on the APB bus. The processor programs the Control registers to begin transactions, and can read progress and error status from the Status Register. FIFO access is normally performed via the APB_DMA module.

22.5.3 FIR Mode Programming

1. Set the CLK_SOURCE_VFIR for 48MHz and enable the UART_B Clock
2. Program IR.TX and IR.RX pin mux settings to select IR (UART B)
3. Set the VFIR.CTL Mode to binary 100 (FIR)
4. FIR mode is Half-Duplex. Set the VFIR.CTL Transmit bit to '1' for transmit, '0' for receive
5. Read the VFIR.STS register to clear interrupts
6. Program the APB_DMA to transfer data to/from the VFIR module
7. Enable the DMA bit in the VFIR.CTL register
8. Enable interrupts as needed
9. Set the GO bit in the VFIR.CTL register to start the action
10. Wait for DONE status or interrupt which indicates that the transaction has completed
11. Check for Errors when reading the VFIR.STS register
12. Clear status bits by writing a '1' to them before starting next packet.
13. For receive operations, the Incoming Frame Data Length (IFDL) register will contain the number of bytes received. DMA should have read this many bytes from the FIFO.

22.5.4 VFIR Mode Programming

1. Set the CLK_SOURCE_VFIR for 48MHz and enable the UART_B Clock
2. Program IR.TX and IR.RX pin mux settings to select IR (UART B)
3. Set the VFIR.CTL Mode to binary 101 (VFIR)
4. FIR mode is Half-Duplex. Set the VFIR.CTL Transmit bit to '1' for transmit, '0' for receive
5. Read the VFIR.STS register to clear interrupts
6. Program the APB_DMA to transfer data to/from the VFIR module
7. Enable the DMA bit in the VFIR.CTL register
8. Enable interrupts as needed

9. Set the GO bit in the VFIR.CTL register to start the action
10. Wait for DONE status or interrupt which indicates that the transaction has completed
11. Check for Errors when reading the VFIR.STS register
12. Clear status bits by writing a '1' to them before starting next packet.
13. For receive operations, the Incoming Frame Data Length (IFDL) register will contain the number of bytes received. DMA should have read this many bytes from the FIFO.

22.5.5 COM-SLAVE Interface (UART B)

The COM-SLAVE interface consists of a UART (Universal Asynchronous Receiver/Transmitter), which provides serial data synchronization, parallel-to-serial and serial-to-parallel data conversion for both receiver and transmitter sections. Synchronization for the serial data stream is accomplished by adding start and stop bits to the transmit data to form a data character. Data integrity is accomplished by attaching a parity bit to the data character. The parity bit can be checked by the receiver for any transmission bit errors.

The COM-SLAVE interface is fully programmable by an 8-bit CPU interface. The registers are 32-bit Word aligned. The interface supports word lengths from five to eight bits, an optional parity bit and one or two stop bits. If enabled, the parity can be odd, even or forced to a defined state. Interrupts can be generated from any of ten sources.

The UART supports both 16450 and 16550 compatible modes. Default mode is the 16450. The 16550 mode, which provides independent 16-byte FIFOs for transmit and receive, is selected via the FIFO control register. It also includes a 16-bit programmable baud rate generator and an 8-bit Scratch register, eight modem control lines and two DMA handshake lines that are used to indicate when the FIFOs are ready to transfer data to the CPU.

Table 70 Serial Bus Interface Signals for UART B

Signal Name	Direction	Description
UB3_RXD	I	Receiver Data Port
UB3_TXD	O	Transmit Data Port

22.5.6 SIR Pulse Encoder/Decoder

The UART transmit (TX) data is passed through this module prior to being muxed out to the IR transmitter. This module converts transmitted zeros into a 3/16 Return-To-Zero (RZ) pulse.

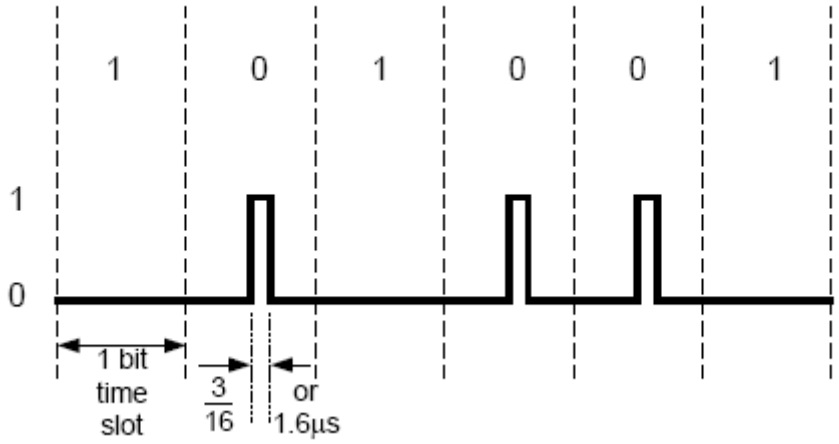
Similarly, received (RX) data is bit-synchronized using the 16X baud clock and the original serial data recovered from the IR bit stream. This data is then sent to the UART for reception.

The signal generated in SIR is as follows:

- On logic '1' the LED is off.
- On logic '0' a pulse is created starting the center of the bit time and lasting 3/16 of bit time period or 1.6µs (3/16 bit times at 115.2 kbps) depending on the current settings.

The figure below displays the output for the input 101001 (in sending order) in SIR encoding.

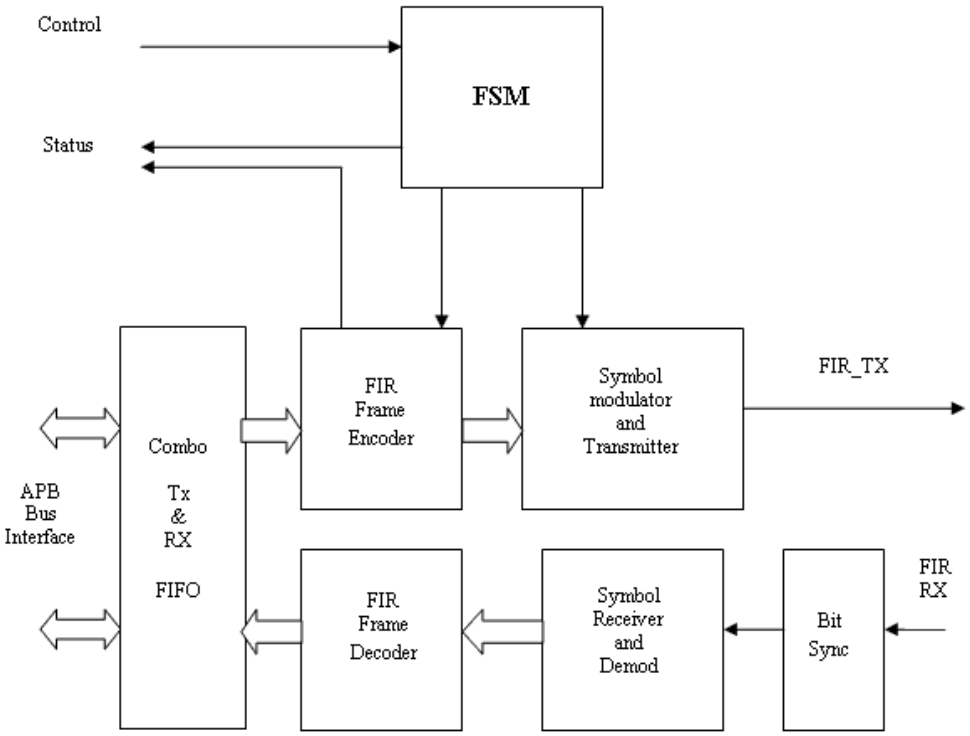
Figure 44 Output in Sending Order in SIR Encoding



22.5.7 FIR Encoder/Decoder

The Figure below shows the micro-architecture for FIR and VFIR modes. Although the architecture is the same, the FIR and VFIR encoding schemes are very different.

Figure 45 Architecture for FIR and VFIR Modes



The Frame Level Encoder calculates CRC for the sent data, performs bit stuffing, creates start and stop sequences and sends data down the line to the Data Shift Register. The Frame Level Decoder calculates the CRC on incoming data, performs bit de-stuffing, compares the calculated CRC to the received one and reports any errors detected in the process.

The 1-symbol modulator creates the signal to be sent to the IR transceiver bit by bit as received from the UART in SIR, 2-bit signals in FIR, and 3-bit signals in VFIR mode.

FIR mode uses a Pulse Position Modulation code called 4PPM. The encoding sends one symbol every four time slices called "chips". Because there are four unique chip positions within each symbol in 4PPM, four independent symbols exist in which only one chip is logically a "one" while all other chips are logically a "zero." These four unique symbols are defined to be the only legal data symbols (DD) allowed in 4PPM. Each DD represents two bits of payload data, so that a byte of payload data can be represented by four DDs in sequence. The table below defines the chip pattern representation of the four unique DDs defined for 4PPM.

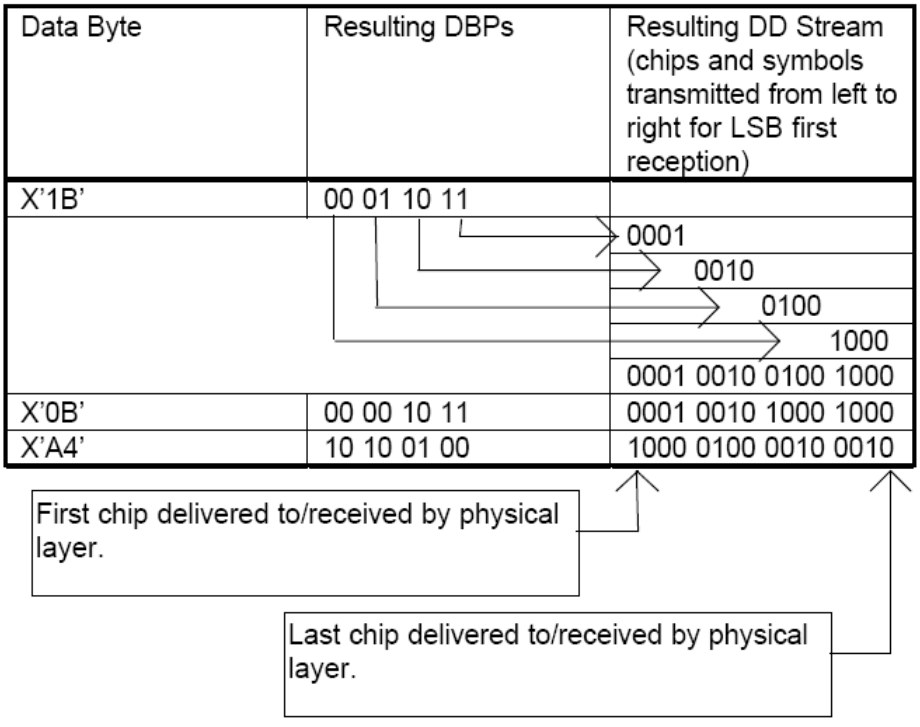
Table 71 Chip Pattern

Data Bit Pair	4PPM Data Symbol
00	1000
01	0100
10	0010
11	0001

Logical "1" represents a chip duration when the transmitting LED is emitting light, while logical "0" represents a chip duration when the LED is off. Data encoding for transmission is done LSB first.

The figure below shows how various data bytes would be represented after encoding for transmission. In these examples transmission time increases from left to right so that chips and symbols farthest to the left are transmitted first.

Figure 46 Various Data Bytes After Encoding for transmission



22.5.8 PPM Packet Format

For FIR 4PPM packets the data rate is 4 Megabits per second and the signaling rate is 8.0 Mchips per second, where a chip is the smallest element of IrDA signaling. The following packet format is defined:

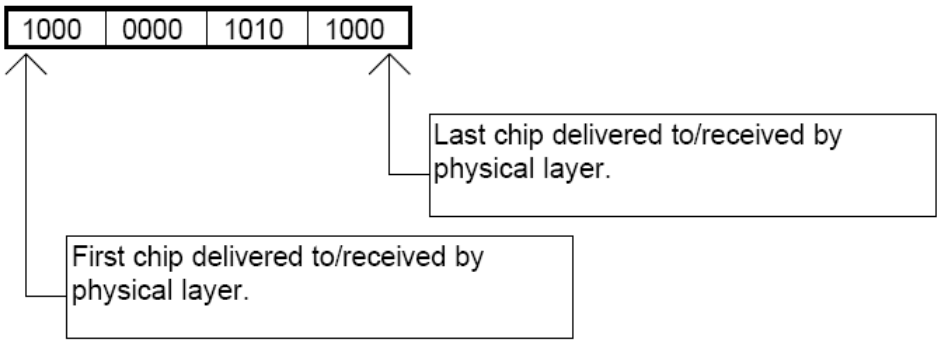
In this packet format, the payload data is encoded as described in the 4PPM encoding above, and the encoded symbols reside in the DD field. Maximum packet length is negotiated by the same mechanism as for the slower rates. The preamble field (PA) is used by the receiver to establish phase lock. During PA, the receiver begins to search for the start flag (STA) to establish symbol synchronization. If STA is received correctly, the receiver can begin to interpret the data symbols in the DD field. The receiver continues to receive and interpret data until the stop flag (STO) is recognized. STO indicates the end of a frame. The chip patterns and symbols for PA, STA, FCS field, and STO are defined below. Only complete packets that contain the entire format defined above are guaranteed to be decoded at the receiver (note that, as for the lower rates, the information field, I, may be of zero length).

The 4PPM data encoding described above defines only the legal encoded payload data symbols. All other4 chip combinations are by definition illegal symbols for encoded payload data. Some of these illegal symbols are used in the definition of the preamble, start flag, and stop flag fields because they are unambiguously not data.

22.5.8.1 Preamble Field Definition

The preamble field (PA) consists of exactly sixteen repeated transmissions of the following stream of symbols. In the PA field, transmission time increases from left to right so that chips and symbols on the left are transmitted first. See figure below.

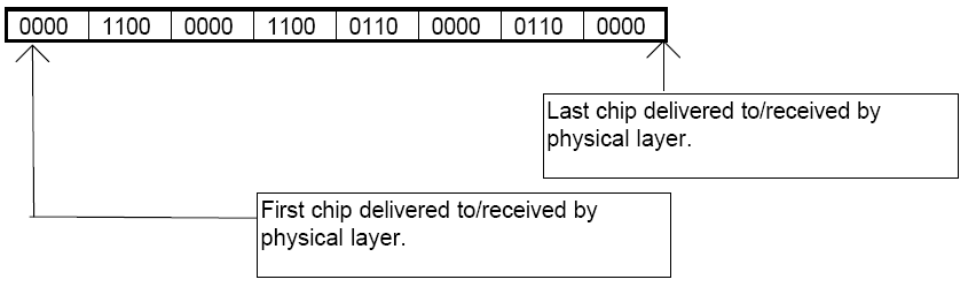
Figure 47 Preamble Field



22.5.8.2 Start Flag Definition

The start flag (STA) consists of exactly one transmission of the following stream of symbols. In the STA field, transmission time increases from left to right so that chips and symbols on the left are transmitted first. See figure below.

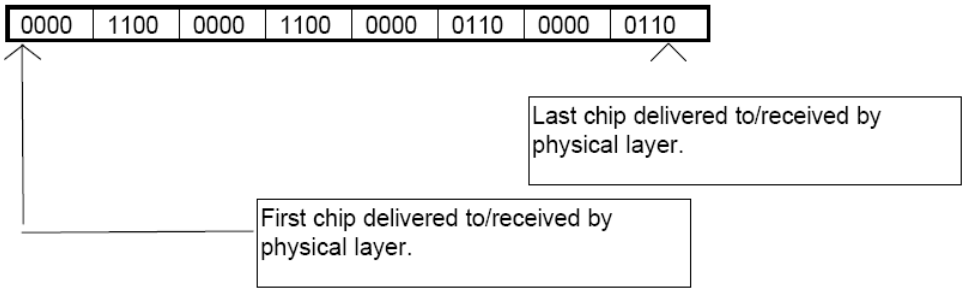
Figure 48 Start Flag



22.5.8.3 Stop Flag Definition

The stop flag (STO) consists of exactly one transmission of the following stream of symbols. In the STO field, transmission time increases from left to right so that chips and symbols on the left are transmitted first. See figure below.

Figure 49 Stop Flag



22.5.8.4 Frame Check Sequence Field Definition

Frame check sequence (FCS) field is a 32 bit field that contains a cyclic redundancy check (CRC) value. The CRC is a calculated, payload data dependent field, calculated before 4 PPM encoding. It consists of the 4PPM encoded data resulting from the IEEE 802 CRC32 algorithm for cyclic redundancy check as applied to the payload data contained in the packet. The CRC32 polynomial is defined as follows:

$$CRC(x) = x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x + 1$$

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.

22.5.8.5 Aborted Packets

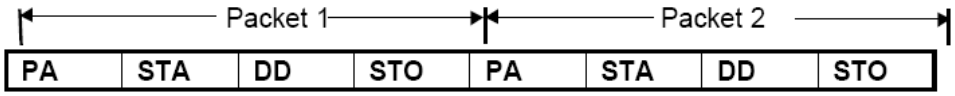
Receivers may only accept packets that have valid STA, DD, FCS, and STO fields as defined in the PPM Packet Format section. The PA field need not be valid in the received packet. All other packets are aborted and ignored.

Any packet may be aborted at any time after a valid STA but before transmission of a complete STO flag by two or more repeated transmissions of the illegal symbol "0000." Also, any packet may be aborted at any time after a valid STA by reception of any illegal symbol which is not part of a valid STO field.

22.5.8.6 Back to Back Packet Transmission

Back to back, or "brick-walled" packets are allowed, but each packet must be complete (i.e., containing PA, STA, DD and STO fields). Brick-walled packets are illustrated in the figure below.

Figure 50 Back-to-Back Packet Transmission



22.5.9 VFIR 16.0 Megabits per Second Rate

The 16.0 Megabits per second data transmission incorporates a new modulation code HHH (1,13) - low duty cycle, rate 2/3, (d,k) = (1, 13) run-length limited (RLL) code to achieve the specified data rate from a 24MHz reference clock. The HHH(1,13) code guarantees for at least one empty chip and at most 13 empty chips between chips containing pulses in the transmitted IR signal. The data transmission packet/ frame is based on the 4.0 Megabits per second frame format with modifications introduced where necessary to accommodate the requirements that are specific to the new modulation code. The system includes a simple scrambling/descrambling scheme to randomize the duty cycle statistics. The signaling rate of the 16.0 Megabits per second data rate is 24.0 Mchips per second, where a chip is the smallest element of IrDA signaling.

22.5.10 HHH (1, 13) Modulation Code

The HHH (1, 13) modulation code has the following salient features:

- Code Rate: 2/3 ,
- Maximal Duty Cycle: 1/3 (~33%) ,
- Average Duty Cycle: ~26% ,
- Minimal Duty Cycle: 1/12 (~8.3%) ,
- Run-Length Constraints: (d, k) = (1, 13) ,
- Longest Run of '10's: yyy'000'101'010'101'000'yyy ,
- Chip Rate @ Data Rate 16 Megabits per second: 24 Mchips per second ,
- System Clock @ Data Rate 16 Megabits per second: N 12 MHz (where N ≥ 4)

The HHH(1,13) code is a Run Length Limited (RLL) code that provides both power efficiency and bandwidth efficiency at the high data rate. The signaling rate of the code is 24 Mchips per second allowing a rise and fall time of 19 ns. LED on time is further improved by having a 26% average duty cycle for random data. The lower duty cycle is achieved by scrambling the incoming data stream. The run length constraints (d, k) = (1, 13) ensure an inactive chip after each active chip, i.e. only single-chip-width pulses occur. This feature allows a source or a receiver to exhibit a long tail property. To take full advantage of the d = 1 feature of HHH(1, 13) in strong signal conditions, clock and data recovery circuitry ignores the level of the chip following an active chip and assumes these chips are inactive. The modulation code is enhanced with simple frame-synchronized scrambler/descrambler mechanisms as defined and described in later in this chapter.

22.5.10.1 HHH Data Encoding Definition

The encoding definition of the HHH (1, 13) code is provided by a state transition table described as follows:

Specific data pair D ≡ D* = (d1, d2) arriving at the encoder input is first associated with a corresponding next state N ≡N*. This occurs as soon as the data D* have advanced into the positions of the internal data bits B1 = (b1, b2), i.e., when (b1, b2, b3, b4, b5, b6) is identical to (d1, d2, x, x, x, x). In a second step, during the next encoding cycle, the state S takes on the value of N*, i.e., S ≡ S* ← N* so that S is now associated with (d1, d2). In the same cycle, the inner codeword C ≡ C* now carrying the information of D* is computed. Thus, referring the Figure below, a given internal input vector (b1, b2, b3, b4, b5, b6) associates the bits (b1, b2) with the next state N and a given state S associates the data pair ahead of (b1, b2) to the output C. In other words, the pair-wise values for N and C as listed in the Figure below are not associated with the same input data pair.

Encoder initialization: The state S = (s1, s2, s3) = (1, 0, 0) is also used as the initial state of the encoder, i.e., denoting with (α, β), the first pair of data bits to be encoded, the state S is forced to take on the value (1, 0, 0) when the bits (α, β) have advanced into the encoding circuits such that the internal inputs B1 = (b1, b2) is identical to (α, β)

Figure 51 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
0 0 0	000/010	001/010	010/010	111/010	111/001	111/010	011/010	011/010
0 0 1	000/001	001/001	100/001	100/010	100/010	100/010	100/010	100/010
0 1 0	000/100	001/100	010/100	111/100	111/101	111/100	011/100	011/100
0 1 1	000/101	001/101	100/101	100/100	100/100	100/100	100/100	100/100
1 0 0 1)	000/000	001/000	010/000	011/000	011/000	011/000	011/000	011/000
1 1 1	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.

22.5.10.2 HHH (1, 13) Packet Format

The packet format for 16.0 Megabits per second HHH(1,13) has the following form:

PREAMBLE (PA)	START (STA)	IrLAP Frame	CRC	Flush Byte (FB)	STOP (STO)	NULL

The payload data is encoded as described in the HHH (1,13) encoding above, and the encoded symbols reside in the IrLAP Frame field. The preamble field (PA) is used by the receiver to establish phase lock. During PA, the receiver begins to search for the start flag (STA) to establish symbol synchronization. If STA is received correctly, the receiver can begin to interpret the data symbols in the IrLAP Frame field. The receiver continues to receive and interpret data until the stop flag (STO) is recognized. STO indicates the end of a frame. The chip patterns and symbols for PA, STA, CRC field, and STO are defined below.

Only complete packets that contain the entire format defined above are guaranteed to be decoded at the receiver (note that, as for the lower rates, the information field, I, that is part of the IrLAP field, may be of zero length).

The 16.0 Megabits per second packet contains several fields for the purposes of clock recovery, synchronization and data transmission. In concept, the packet format is similar to that used in 4.0 Megabits per second, however, there are specific controller elements like clock recovery, synchronization and encoding/decoding circuits that need to be implemented specifically for 16.0 Megabits per second data rate.

22.5.10.3 Preamble Field Definition

The transmitted PREAMBLE (PA) is constructed by concatenating ten times (10) the 24-chip (1 µs) PREAMBLE PERIOD (PP), where

PP = '100'010'010'001'001'001'000'100'

to form the complete 240-chip (10 µs) preamble

PA = 'PP'PP'PP'PP'PP'PP'PP'PP'PP'PP'

The left-most/right-most chip of PP and PA, respectively, is transmitted first/last and a '1' in PP means an active chip (pulse) and a '0' means an empty chip (no pulse).

22.5.10.4 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.

22.5.10.5 IrLAP Frame

The structure remains unchanged from that defined in the IrLAP Specification, Version 1.1. The content of the IrLAP frame is first scrambled and then encoded with HHH(1,13). Note that the 32 CRC bits for the IrLAP frame are calculated before the IrLAP frame is scrambled. For reference, the IrLAP frame as the following structure:

| Address (8 bits) | Control (8 bits) | Information (M times 8 bits) |

22.5.10.6 CRC

Computation remains unchanged from the 32-bit CRC defined for the 4 Megabits per second data rate. The content of the CRC field is first scrambled with the scheme described later and then encoded with HHH(1, 13) as described previously. Note that the 32 CRC bits for the IrLAP frame are calculated before the IrLAP frame is scrambled. The transmitted CRC field is a 48-chip (2 s) sequence.

22.5.10.7 FLUSH BYTE (FB)

The Flush Byte (FB) is the 8-bit sequence:

FB = '00'00'00'00'.

These 8 bits are not scrambled but directly sent to the HHH(1,13) encoder. The transmitted FB field is a 12-chip (0.5 s) sequence. Note that the FB field is required to enable complete decoding of the CRC field. The flush byte denotes the end of the main body. Since the flush byte is not scrambled, a well balanced HHH(1,13) sequence precedes the STOP delimiter. This sequence would be equivalent to the UART "Break" command.

22.5.10.8 STOP (STO)

The transmitted STOP (STO) delimiter is the 48-chip (2 µs) sequence

STO = '001'001'010'101'001'000'100'000'100'101'010'100'100'000'100'000'.

The left-most/right-most chip of STO is transmitted first/last and a '1' in STO means an active chip (pulse) and a '0' means an empty chip (no pulse). As in the Start Flag delimiter, the Stop flag also contains a subsequence '1001010101001' that violates the HHH (1,13) code. This subsequence also occurs twice in the Stop Flag delimiter.

22.5.10.9 NULL Sequence

The transmitted NULL sequence is the 24-chip (1 µs) sequence

NULL = '000'000'000'000'000'000'000'000'.

The NULL field is a new field for the purpose of providing an HHH(1,13) code pattern violation that permits terminating reception of the packet in the event that the STO field is not recognized. The left-most/right-most chip in NULL is transmitted first/last and all chips of NULL are empty chips (no pulses). The NULL field increases the probability that the packet is terminated close to the STOP flag. The NULL field also reduces the probability that two back to back packets are interpreted as a single packet, should the STOP flag delimiter of the first packet be missed.

22.5.10.10 Scrambling and Descrambling Functions

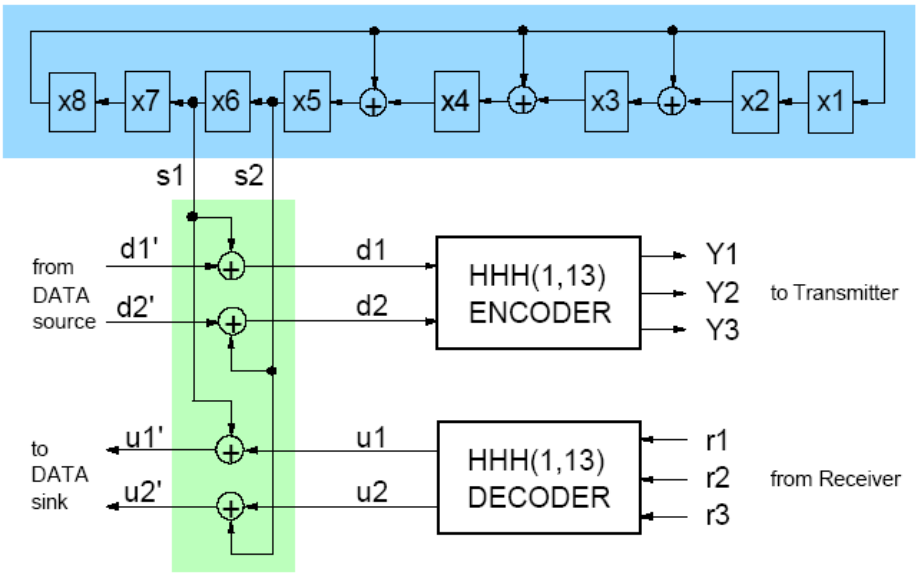
It is advantageous to enhance the encoder/decoder system with simple scrambler/descrambler functions.

The primitive polynomial: $x^8 \oplus x^4 \oplus x^3 \oplus x^2 \oplus 1$,

where ⊕ indicates a modulo-2 addition or, equivalently, a logic exclusive OR (XOR) operation The operations of the proposed scrambling and descrambling functions are performed according to the principles of frame synchronized scrambling/descrambling (FSS) mechanisms. The scrambling/descrambling scheme is shown in the Figure below. The linear feedback shift register (LFSR) produces a maximum-length pseudo-random sequence with period 255. The output of register cell x6 shown in the figure is defined to be the equivalent serial output of the LFSR.

The modulo-2 adders shown in the figure below correspond to logic XOR (exclusive OR) gates. During transmission, each new pair of source bits (d1', d2') is XOR-ed with a new pair of scrambling bits (s1, s2) to produce the scrambled data bit pair (d1, d2) entering the encoder. Similarly, during reception, each new pair of decoded bits (u1, u2) is XOR-ed with a new pair of descrambling bits (s1, s2) to produce the descrambled user bit pair (u1', u2') that is sent to the data sink. A scrambling/descrambling cycle has duration 3T seconds where T = 41.7 ns is the chip period.

Figure 52 Scrambling / Descrambling Scheme



22.5.10.11 Effects and Limits of Scrambling/Descrambling

By enhancing the system with scrambling/descrambling functions during data transmission/reception, one achieves generally better duty cycle statistics in the HHH(1,13) coded channel chip stream; the resulting duty cycle converges towards the average duty cycle of the code (≈ 26%) for typical payload data. Scrambling can greatly reduce the probability of occurrence of worst-case patterns.

22.5.10.12 Aborted Packets

Receivers may only accept packets that have valid STA, IrLAP frame, CRC, and STO fields as defined in the Packet Format section. The PA field need not be valid in the received packet. All other packets are aborted and ignored.

22.5.10.13 Back to Back Packet Transmission

Back to back, or "brick-walled" packets are allowed, but each packet must be complete (i.e., containing PA, STA, IrLAP Frame, CRC and STO fields).

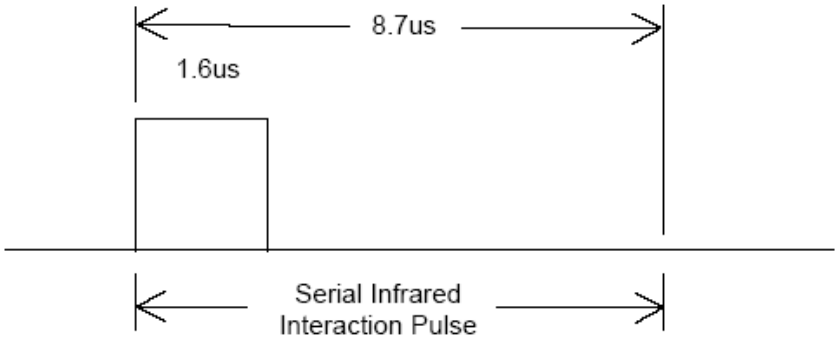
22.5.11 SIR - Serial Interaction Pulse

In order to guarantee non-disruptive coexistence with slower (up to 115.2 Kilobits per second) systems, once a higher speed (above 115.2 Kilobits per second) connection has been established, the higher speed system must emit a Serial infrared Interaction Pulse (SIP) at least once every 500 ms as long as the connection lasts to quiet slower systems that might interfere

with the link. The pulse can be transmitted immediately after a packet has been transmitted. Initiation of this pulse will be performed by software by setting a bit in the VFIR Control register.

The pulse is shown below.

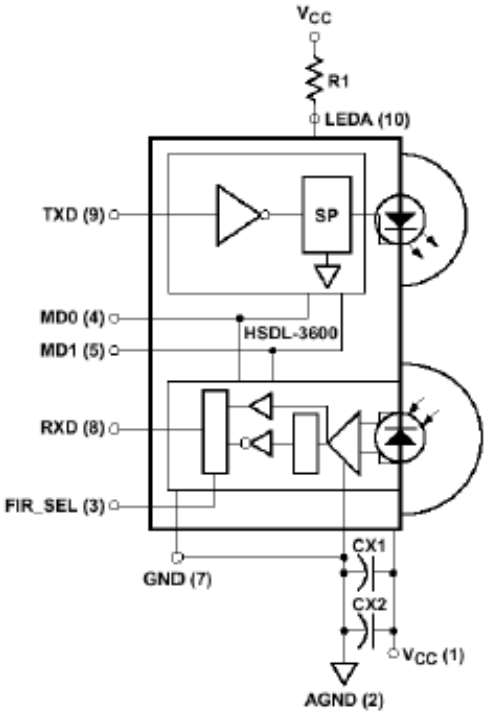
Figure 53 SIP



22.5.11.1 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 will all venders. Other transceiver pins are not directly supported, although they could be provided using GPIO signals. Refer to the drawing below for typical interface. TXD and RXD are VFIR module pins. MD and FIR_SEL pins would be GPIO. MD pins are for transmit power. FIR_SEL controls gain of receiver, and should be set to zero for SIR mode.

Figure 54 Typical Interface



22.6 UART Registers

22.6.1 UART_THR_DLAB_0_0

This UART is based on the 16550 industry standard Universal Asynchronous Receiver/Transmitter (UART), with enhancements to support Autobaud detection and End-of-Received Data timeout detection.

The UART supports a device clock of up to 72 MHz, for a maximum baud rate of 4.5 Megabits per second.

The THR, RBR and DLL registers all occupy the same address space.

- The Transmitter Holding Register (THR) is Write-Only, and can be accessed if the LSR.DLAB bit is clear.
- The Receiver Buffer Register (RBR) is Read-Only, and can be accessed if the LSR.DLAB bit is clear.
- The Divisor Latch LSByte Register (DLL) is Read/Write and can be accessed if the LSR.DLAB bit is set.

UART Transmit Holding Register

Offset: 000h | Read/Write: R/W | Reset: 0b00000000

Bit	R/W	Reset	Description
7:0	RO	0x0	THR_A: Transmit holding register, holds the character to be transmitted by the UART. In FIFO mode, a write to this FIFO places the data at the end of the FIFO.
7:0	RO	X	RBR_A: Receive Buffer Register. Rx Data read from here.
7:0	RO	0x0	DLL_A: Divisor Latch LSB (low 8 bits of 16-bit Baud Divisor)

22.6.2 UART_IER_DLAB_0_0

The IER and DLH registers occupy the same address space, selected by the LSR.DLAB bit. The Interrupt Enable Register (IER) is selected if the LSR.DLAB bit is clear. The Divisor Latch MSByte register (DLM) is selected if the LSR.DLAB bit is set. The DLM register holds the upper 8 bits of the 16-bit Baud Divisor (16x).

UART Interrupt Enable por=0x00000000

RX_TIMEOUT occurs when data has been sitting in the Rx FIFO for more than 4 character times without being read because there is not enough data to reach the trigger level. Interrupt needed to handle this case so that all data is received in a timely manner. Note that this normally occurs at the end of an incoming data stream.

EORD (End of Receive Data) Interrupt occurs when the receiver detects that data stops coming in for more than 4 character times. Useful for determining that the sending device has completed sending all its data. EORD timeout will not occur if the receiving data stream is stopped because of hardware handshaking. To clear the EORD timeout interrupt you must DISABLE the EORD interrupt enable (IE_EORD)

Interrupt Enable and Divisor Latch MSByte Registers

Offset: 004h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7:6	0x0	N_A: Reserved
5	0x0	IE_EORD: Interrupt Enable for End of Received Data 1 = Enable 0 = DISABLE 1 = ENABLE
4	0x0	IE_RX_TIMEOUT: Interrupt Enable for Rx FIFO timeout 1 = Enable 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
3	0x0	IE_MSI: Interrupt Enable for Modem Status Interrupt 0 = DISABLE 1 = ENABLE **NOTE:** Modem Status Interrupt (IE_MSI) should not be enabled when the modem is used in 4-pin or 2-pin configurations. If enabled in these configurations phantom interrupts may be generated.
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

22.6.3 UART_IIR_FCR_0

The FCR and IIR registers occupy the same address space. The FIFO Control Register (FCR) is Write-Only. The Interrupt Identification Register (IIR) is Read-Only

UART FIFO Control Register por=0x00000000

The DMA can run in one of two modes:

- If Mode0 is selected (NO_CHANGE) the Rx DMA request will go active whenever the Rx FIFO is not empty. Only single byte transactions can be performed in this mode. If the FIFO is not enabled, Mode0 MUST be used.
- If Mode1 is selected (CHANGE) the Rx DMA request will go active whenever the Rx FIFO trigger level is reached.

For best performance, always enable the FIFO and select DMA Mode 1.

For RX_TRIG the FIFO_COUNT references the number of bytes in the receive FIFO

For TX_TRIG the FIFO_COUNT references the number of empty bytes in the transmit FIFO. For example, FIFO_COUNT_GREATER_16 means that there are at least 16 empty byte slots in the TX_FIFO

UART Interrupt ID Register por=0x00000001

The Interrupt ID field indicates the current highest priority interrupt. If more than one interrupt is pending the one with the highest priority will be shown.

The table below shows the encoding. Priority flows from top (highest) to bottom (lowest).

Table 72 Interrupt ID encoding

IIR[3:0]	Priority Level
0001	no interrupt
0110	Overrun Error, Parity Error, Framing Error, Break
0100	Receiver Data Available
1100	rx_timeout_intr
1000	eord_timeout_intr
0010	transmitter holding register empty
0000	modem_status interrupt

UART FIFO Control and Interrupt Identification Registers

Offset: 008h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7:6	X	EN_FIFO: FIFO Mode Status 0=16450 mode(no FIFO), 1 = 16550 mode (FIFO) 1 = MODE_16550 0 = MODE_16450
7:6	0x0	RX_TRIG: FIFO_COUNT_GREATER_12 deprecated 0 = FIFO_COUNT_GREATER_1 1 = FIFO_COUNT_GREATER_4 2 = FIFO_COUNT_GREATER_8 3 = FIFO_COUNT_GREATER_16 3 = FIFO_COUNT_GREATER_12
5:4	X	N_A: Reserved
5:4	0x0	TX_TRIG: FIFO_COUNT_GREATER_12 deprecated 0 = FIFO_COUNT_GREATER_16 1 = FIFO_COUNT_GREATER_8 2 = FIFO_COUNT_GREATER_4 3 = FIFO_COUNT_GREATER_1 0 = FIFO_COUNT_GREATER_12
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 FIFO. This bit should be enabled 0 = DISABLE 1 = ENABLE

22.6.4 UART_LCR_0

UART Line Control Register

Offset: 00ch | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	DLAB: Divisor Latch Access Bit (set to allow programming of the DLH, DLM Divisors) 0 = DISABLE 1 = ENABLE
6	0x0	SET_B: Set BREAK condition -- Transmitter will send all zeroes to indicate BREAK 0 = NO_BREAK 1 = BREAK
5	0x0	SET_P: Set (force) parity to value in LCR [4] 0 = NO_PARITY 1 = PARITY
4	0x0	EVEN: Even parity format. There will always be an even number of 1s in the binary representation (PAR = 1) 0 = DISABLE 1 = ENABLE
3	0x0	PAR: 0 = No parity sent 0 = NO_PARITY 1 = PARITY
2	0x0	STOP: 0 = Transmit 1 stop bit, 1 = Transmit 2 stop bits (receiver always checks for 1 stop bit) 0 = DISABLE 1 = ENABLE
1:0	0x0	WD_SIZE: 3=Word length of 8 0 = WORD_LENGTH_5 1 = WORD_LENGTH_6 2 = WORD_LENGTH_7 3 = WORD_LENGTH_8

22.6.5 UART_MCR_0

The RTS and CTS pins are used for hardware handshaking with and external serial device. RTS (Request-To-Send) informs the device that the UART is ready to accept data. CTS (Clear-To-Send) comes from the RTS of the external device and informs us that it is OK to send data.

The RTS pin can be controlled manually with the RTS bit in the MCR, or automatically by setting the RTS_EN bit.

If RTS_EN is set, the RTS pin will automatically remove the Request-To-Send when the FIFO reaches the trigger level.

If the CTS_EN bit is set, the UART transmitter will stop transmitting when the Clear-To-Send input is taken away.

UART Modem Control Register

Offset: 010h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	N_A: Reserved
6	0x0	RTS_EN: 1 = Enable RTS Hardware Flow Control 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
5	0x0	CTS_EN: 1 = Enable CTS Hardware Flow Control 0 = DISABLE 1 = ENABLE
4	0x0	LOOPBK: 1 = enable internal loop back of Serial Out to In 0 = DISABLE 1 = ENABLE
3	0x0	OUT2: nOUT2 (Not Used) 0 = DISABLE 1 = ENABLE
2	0x0	OUT1: nOUT1 (Not Used) 0 = DISABLE 1 = ENABLE
1	0x0	RTS: 0 = Force RTS to high if RTS HW flow control not enabled 0 = FORCE_RTS_HI 1 = FORCE_RTS_LOW
0	0x0	DTR: 1 = Force DTR to high 0 = FORCE_DTR_HI 1 = FORCE_DTR_LOW

22.6.6 UART_LSR_0

UART Line Status Register

Offset: 014h | Read/Write: RO | Reset: 0bxxxxxxxx

Bit	Reset	Description
7	X	FIFOE: 1 = Receive FIFO Error 0 = NO_ERR 1 = ERR
6	X	TMTY: Transmit Shift Reg 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 Errpr 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

22.6.7 UART_MSR_0

UART Modem Status Register

Offset: 018h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	CD: State of Carrier detect pin 0 = DISABLE 1 = ENABLE
6	0x0	RI: State of Ring Indicator pin 0 = DISABLE 1 = ENABLE
5	0x0	DSR: State of Data set ready pin 0 = DISABLE 1 = ENABLE
4	0x0	CTS: State of Clear to send pin 0 = DISABLE 1 = ENABLE
3	0x0	DCD: Change (Delta) in CD state detected 0 = DISABLE 1 = ENABLE
2	0x0	DRI: Change (Delta) in RI state detected 0 = DISABLE 1 = ENABLE
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

22.6.8 UART_SPR_0

UART Scratch Pad Register

Offset: 01ch | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7:0	0x0	SPR_A: Scratchpad register (not used internally)

22.6.9 UART_IRDA_CSR_0

Bits [7:6] control the InfraRed (SIR) encoding. If enabled each zero bit sent will be transmitted as a short IR pulse. No pulse is sent during idle or '1' data bits. The IR pulse can be either 3/16 or 4/16 of a normal bit time.

The low 4 bits control signal polarity and apply whether or not SIR coding is enabled.

UART IrDA Pulse Coding CSR Register

Offset: 020h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	SIR_A: 1 = Enable SIR coder 0 = Disable SIR coder

Bit	Reset	Description
		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
5:4	0x0	N_A: Reserved = 0
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

22.6.10 UART_ASR_0

The UART has auto-Baud sensing logic which can measure the width of the start bit of incoming data to determine baud rate. For this to work the incoming character must have it's LSB set. A <cr> works well (0x0d). The logic will use the UART device clock to count how wide the start pulse is and the BUSY bit will be set when the sensing is complete. The value in {RX_RATE_SENSE_H,RX_RATE_SENSE_L} should be divided by 16 with Round-to-Nearest, and the resulting value loaded into the DLM,DLL register pair to set the Baud Clock to 16X the Baud Rate.

UART Auto Sense Baud Register

Offset: 03ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

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 dont 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
29:16	0x0	N_A: Reserved = 0
15:8	0x0	RX_RATE_SENSE_H: Shows the bits [15:8] of the count of clock edges between two successive clock edges.
7:0	0x0	RX_RATE_SENSE_L: Shows the bits[7:0] of the count of clock edges between two successive clock edges.

22.7 VFIR Registers

22.7.1 VFIR_CTL_0

VFIR Control register is used

- to enable/disable entire IRDA module
- to configure the controller in FIR mode or VFIR mode or SIR or UART mode
- to configure either in Transmit mode or Receive mode to invert the polarity of Tx or Rx line
- to enable/disable DMA
- to know the attention levels of Transmit fifo and Receive fifo
- to start Transmit or Receive operation
- to select the Frequency
- to clear Transmitter fifo/Receiver Fifo

VFIR Control Register

Offset: 000h | Read/Write: R/W | Reset: 0b00000000000000000000000000100000

Bit	Reset	Description
31	0x0	GLOBAL_ENABLE: 0 = Entire IRDA module is disabled 0 = DISABLE 1 = ENABLE
30	0x0	GO: Set to 1 to start transmit or receive operation, self-clearing 0 = STOP 1 = START
29	0x0	AUTO_RESTART: Set to 1 to restart another operation when previous is done 0 = STOP 1 = RE_START
24	0x0	FORCE_BAD_CRC: Debugging bit to force a CRC error 0 = DISABLE 1 = ENABLE
22	0x0	TX_FIFO_CLR: Clear the Transmitter FIFO 0 = DISABLE 1 = ENABLE
21:20	0x0	TX_ATN_LVL: 00: when not full 17 = slots_empty_12 16 = 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 17 = slots_full_12 16 = 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

Bit	Reset	Description
		1 = ENABLE
11	0x0	DMA_EN: DMA Enable Allow requests to go APB_DMA controller. 0 = DISABLE 1 = ENABLE
10	0x0	TX_TERM: of outgoing frame, i.e. the calculated CRC will be sent, followed by the frame stop sequence. The interrupt will be generated if enabled. 0 = ABORT 1 = NORMAL
9	0x0	COUNT_ODATA: When enabled, the controller will end the frame when the Outgoing Frame Data Length (OFDL) count is reached. Use in conjunction with TX_Term above so controller will know when and how to end the frame. 0 = DISABLE 1 = ENABLE
8	0x0	FORCEBRK: break state (zero) regardless of what the transmitter is doing. 0 = TX_ENABLE 1 = TX_DISABLE
7	0x0	NEGATE_RX: Invert polarity on incoming RX pin (applies to all modes) 0 = DISABLE 1 = ENABLE
6	0x0	NEGATE_TX: Invert polarity on outgoing TX pin (applies to all modes) 0 = DISABLE 1 = ENABLE
5:4	0x2	FREQUENCY: The 8 and 24 MHz clocks can be used to save power during transmit. The 72MHz clock can be used in case the 48MHz clock is not available due to system constraints. 0 = F8MHz 1 = F24MHz 16 = F48MHz 17 = F72MHz
3	0x0	TRANSMIT: 0 = Receive (FIR/VFIR modes) 1 = TRANSMIT 0 = RECEIVE
2:0	0x0	MODE: 000: UART use UART B module 273 = RSVD 272 = RSVD1 257 = VFIR 256 = FIR 17 = RSVD2 16 = MIR 1 = SIR 0 = UARTB

22.7.2 VFIR_STS_0

VFIR Status and Interrupt Identification Register is used to know the status of:

- Transmitter/Receiver FIFO Trigger level status
- Transmitter/Receiver FIFO FULL/EMPTY status
- Transmitter/Receiver FIFO empty slots
- whether Transmit or Receive operation is done or not
- whether End of Frame is received or not

- whether there is any CRC error in received data or not
- whether there is any Error in the received data
- Whether the controller is Busy in Transmitting/Receiving the data
- Whether the VFIR interrupt is generated or not. To generate interrupt the corresponding the enable bits has to be set in the VFIR Interrupt Enable Register

VFIR Status and Interrupt Identification Register

Offset: 004h | Read/Write: R/W | Reset: 0b10100000000000000000000000000000

Bit	Reset	Description
31	0x1	TX_FTRIG: Transmitter FIFO Trigger level Reached
30	0x0	TX_FIFO_FULL: Transmitter FIFO is Full
29	0x1	TX_FIFO_EMPTY: Transmitter FIFO is Empty
28:24	0x20	TX_FIFO_EMPTY_COUNT: Number of empty slots (words) in Tx FIFO
23	0x0	RX_FTRIG: Receiver FIFO Trigger level Reached
22	0x0	RX_FIFO_FULL: Receiver FIFO is Full
21	0x0	RX_FIFO_EMPTY: Receiver FIFO is Empty
20:16	0x0	RX_FIFO_FULL_COUNT: Number of words available to Read in Rx FIFO
15	0x0	INTR: Global VFIR interrupt (OR of all FIR/VFIR interrupts)
14	0x0	BUSY: transmitting or receiving data. It is clear when the controller is idle or transmits a Serial infrared Interaction Pulse (SIP)
11	0x0	TX_UNDRN: break. This bit is cleared upon reading from the register.
7	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	0x0	BITSYNC_LOCK: Debug status bit which indicates that receiver is Locked to incoming bit stream.
5	0x0	RX_DETECT: Activity detected on Rx pin This bit is cleared by writing a 1 to this bit.
4	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	0x0	RX_FOVRN: Receiver FIFO overrun error occurred. This bit is cleared by writing a 1 to this bit.
2	0x0	RX_CRC_ERR: CRC check on input data failed. This bit is cleared by writing a 1 to this bit.
1	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	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.

22.7.3 VFIR_IER_0

VFIR Interrupt Enable Register is used to generate VFIR interrupt for different conditions. VFIR Interrupt bit is present in VFIR.

Status and Interrupt Identification Register

For example when the controller finishes the Transmitting/Receiving operation and if IE_DONE is set then Interrupt is generated. Similar is the case for other bits in this register

VFIR Interrupt Enable Register

Offset: 008h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	IE_TX_FTRIG: Enable Interrupt on Transmitter FIFO Trigger level reached 0 = DISABLE 1 = ENABLE
23	0x0	IE_RX_FTRIG: Enable Interrupt on Receiver FIFO Trigger level reached 0 = DISABLE 1 = ENABLE
11	0x0	IE_TX_UNDRN: transmit a break for the receiving side to abort reception. 0 = DISABLE 1 = ENABLE
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

22.7.4 VFIR_OFDL_0

VFIR Outgoing Frame Data Length is used to configure the number of bytes of data to be sent.

VFIR Outgoing Frame Data Length

Offset: 00ch | Read/Write: R/W | Reset: 0b0000000000000000

Bit	Reset	Description
15:0	0x0	OFDL: Set to the length in bytes of the outgoing frame. Length is calculated on the information field only (address, control and data fields only) without the CRC and flags. When in this mode, the amount of data transferred to the controller is counted and when the count is reached the controller finishes sending the frame in normal fashion by transmitting CRC field, STO flag and then the SIP.

22.7.5 VFIR_IFDL_0

VFIR Incoming Frame Data Length gives the information of the data received in bytes.

VFIR Incoming Frame Data Length

Offset: 010h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	IFDL: Running length in bytes of the incoming data stream. At the end of frame reception when the Received End of Frame interrupt occurs, this register reflects the length of the information field received in that frame, not counting the CRC field and other flags. Resets on the start of next frame reception, when a new STA field has been detected.

22.7.6 VFIR_FIFO_0

VFIR FIFO is shared FIFO. Holds the data in Transmitter/Receiver mode

VFIR.FIFO (VFIR FIFO)

Offset: 040h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	DATA: Shared FIFO. When transmitting, this FIFO holds the data to be transmitted. When receiving, this FIFO holds the received data.

23.0 SPI CONTROLLER

23.1 SLINK: SPI Peripheral Interface

The SPI bus is a master-slave based bus that is used to interconnect different devices to a central master.

It allows full-duplex synchronous-serial communication between the controller and external peripheral devices. It consists of 4 signals, SS_N (Chip select), SCK (clock), MOSI (Master data out and Slave data in) and MISO (Slave data out and master data in). The data is transferred on MOSI or MISO based on the data transfer direction on every SCK edge. The receiver always receives the data on the other edge of SCK.

The clock polarity and clock phase determine which edge of the SCK clock will be used to latch the data in the receiver. If the clock polarity is 0, then the SCK signal is 0 when idle and transitions to 1 whenever doing a data transfer. If the clock polarity is 1, then the SCK signal is 1 when idle and transitions to 0 when doing a data transfer. If the clock phase is 0, then the receiver latches the data on the first transition of SCK from idle state. If the clock phase is 1, then the receiver latches the data on the second transition of SCK.

This makes up 4 modes for SPI protocol that are described below:

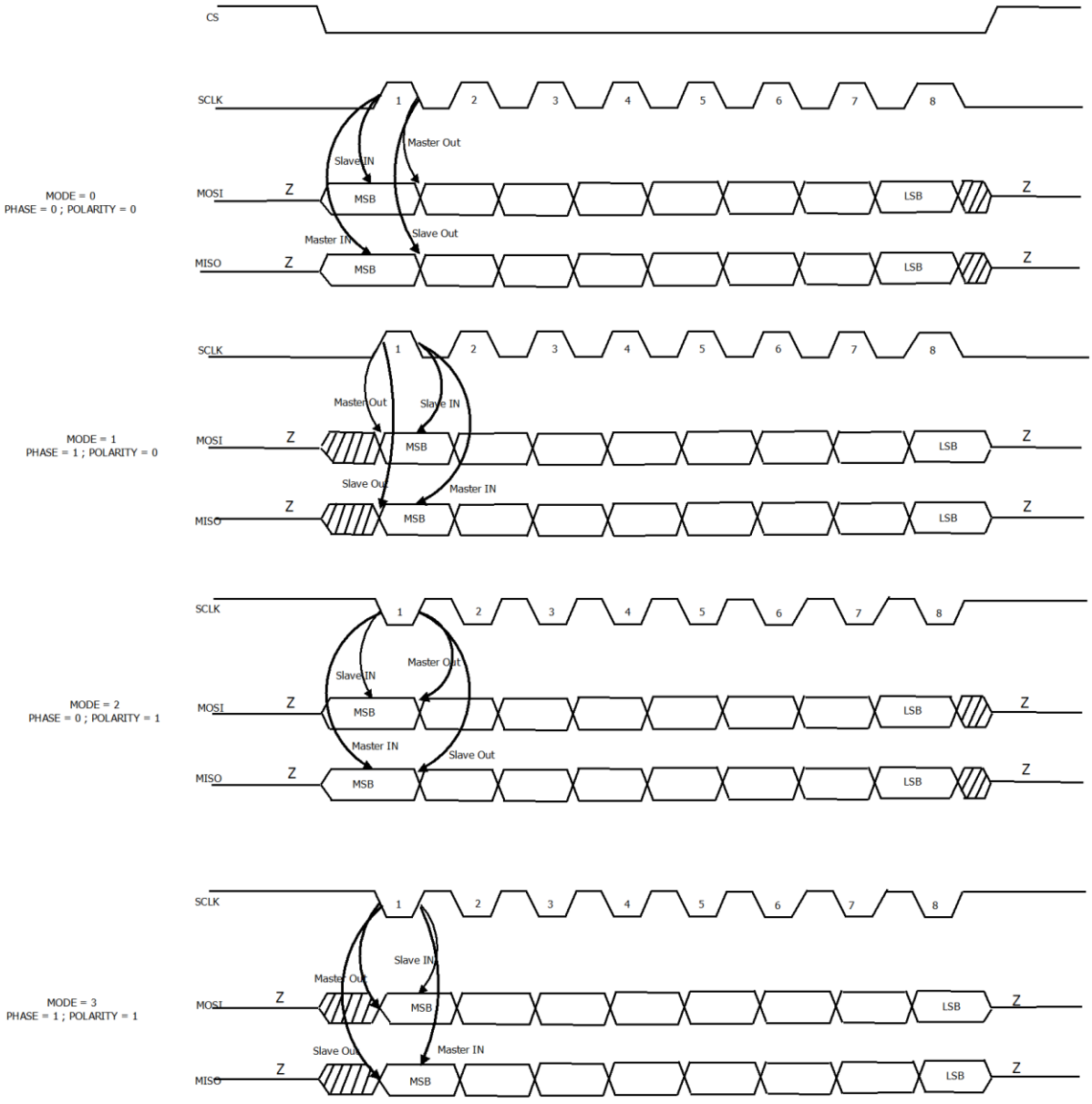
Table 73. SPI Protocols

SPI Mode	Clock Polarity	Clock Phase	Description
0	0	0	Clock is positive polarity and the data is latched on the positive edge of SCK.
1	0	1	Clock is positive polarity and the data is latched on the negative edge of SCK.
2	1	0	Clock is negative polarity and the data is latched on the negative edge of SCK.
3	1	1	Clock is negative polarity and the data is latched on the positive edge of SCK.

In this protocol, data is transferred on every edge of SCK whenever CS is active. CS can be active high or low. The data transfer consists of packets of varying bit lengths from 1 to 32 bits in any direction. Within a packet, the transfer starts with MSB first. The clock can stay in idle state in between different packets. CS can be active for either single packet or multiple packets including transmit and receive.

This protocol is described in the diagram below.

Figure 55. SPI Interface Timing



Features

- Independent RX FIFO and TX FIFO.
- FIFO depth of 32x32-bits.
- Software controlled bit-length from 0 to 31 resulting in packet sizes of 1 to 32 bits.
- Packed mode support for bit-length of 7 (8-bit packet size) and 15 (16-bit packet size).
- SS_N can be selected to be controlled by software, or it can be generated automatically by the hardware on packet boundaries.
- Receive compare mode where the controller listens for a particular pattern on the incoming data before receiving the data in the FIFO.
- Simultaneous receive and transmit supported.
- Maximum transfer rate is 50 Mbits/sec, subject to board electrical characteristics.

23.1.1 Programming Guidelines

There are two basic modes of operation: DMA_EN and GO mode. It is required that software sets up all the parameters in the SBCx_COMMAND and SBCx_COMMAND2 registers before enabling transfers in any of these modes.

23.1.1.1 DMA_EN mode

This mode is enabled by writing 1 to DMA_EN bit in SBCx_COMMAND register. In this mode, SPI controller transmits or receives the number of packets as indicated by the field DMA_BLOCK_SIZE in SBCx_DMA_CTL register

If PACKED bit is set and the BIT_LENGTH is set to 7, then all FIFO words contain 4 packets to transfer (transmit or receive). Packets will be transferred in little-endian format, with packet 0 being in byte 0 of the FIFO and packet 3 in byte 3 of the FIFO.

If PACKED bit is set and the BIT_LENGTH is set to 15, then all FIFO words contain 2 packets to transfer (transmit or receive). Packets will be transferred in little-endian format, with packet 0 being in bits [15-0] of the FIFO and packet 1 in bits [31-16] of the FIFO.

If BIT_LENGTH is set to N, each packet will consist of N + 1 bits. These bits will be transmitted/received in the TX_FIFO/RX_FIFO with the MSB in bit N and LSB in bit 0, following little-endian architecture. Any remaining bits in the FIFO will be ignored by the hardware. Maximum packet length is 32, which can be selected by setting BIT_LENGTH to 31. In this case, all data bits in FIFO contain valid packet data.

DMA request will be generated to APB_DMA in this mode depending on the setting of IE.TXC and IE.RXC. If transmit is enabled, setting IE.TXC to 00 will generate a TX DMA request whenever the TX FIFO has one word of space available (is not full). Setting IE.TXC to 01 will generate a TX DMA request whenever the TX FIFO has 4 words of space available (is empty). If receive is enabled, setting IE.RXC to 00 will generate a RX DMA request whenever the RX FIFO has one word of data available (is not empty). Setting IE.RXC to 01 will generate a RX DMA request whenever the RX FIFO has 4 words of data available (is full).

APB DMA requester number for SPI controller is 12.

23.1.1.2 GO mode

This mode is enabled by writing 1 to GO bit in SBCX_COMMAND register. In this mode, if transmit is enabled, then the SPI controller transmits the data in the TX FIFO until TX FIFO is empty. If receive is enabled, then the SPI controller receives one packet of data in the RX FIFO.

Packed mode is supported in transmit only. If PACKED bit is set and the BIT_LENGTH is set to 7, then all TX FIFO words contain 4 packets to transfer. If PACKED bit is set and the BIT_LENGTH is set to 15, then all TX FIFO words contain 2 packets to transfer.

23.1.1.3 Interrupt Generation

SPI controller generates an interrupt to processor at the end of a transfer or when an error is detected if IE.TXC or IE.RXC bit in SBCX_DMA_CTL register is enabled for transmit and receive modes of operation respectively. This functionality is common to both DMA_EN and GO modes.

If IE.TXC is enabled during transmit, an interrupt is generated whenever RDY or TXF_OVF bit in SBCX_STATUS register is set to 1. During transmit, if software/APB DMA cannot fill the transmit FIFO fast enough, hardware waits for the software to fill up the TX FIFO and an under run condition is never generated. However, if software tries to write to a full TX FIFO, hardware sets TXF_OVF bit as an indication that software attempted to overflow the TX FIFO. Hardware makes sure that the overflowing data is never written to the TX FIFO.

If IE.RXC is enabled during receive, an interrupt is generated whenever RDY or RXF_UNR bit in SBCX_STATUS register is set to 1. During receive, if software/APB DMA cannot read the receive FIFO fast enough, hardware waits for the software to read out the RX FIFO and an overflow condition is never generated. However, if software tries to read from an empty RX

FIFO, hardware sets RXF_UNR bit as an indication that software attempted to under run the RX FIFO. Hardware makes sure that the read actually never goes to an empty RX FIFO.

The interrupt can be cleared by clearing the source of the interrupt. If the interrupt is generated by assertion of RDY, then writing a 1 to RDY bit clears the interrupt. If the interrupt is generated by assertion of TXF_OVF, then writing a 1 to TXF_OVF bit clears the interrupt. If the interrupt is generated by assertion of RXF_UNR, then writing a 1 to RXF_UNR bit clears the interrupt.

SPI Controller interrupt source is bit SPI (bit 7) in secondary interrupt controller.

23.1.1.4 Packet Parameters

Clock signal (SCK) output format: The SCK clock signal output format is controlled by the fields ACTIVE.SCLK and IDLE.SCLK in SBCX_COMMAND register. Both these fields should be programmed with the same value. These fields determine the active clock polarity as indicated below.

Table 74. Active Clock Polarity

ACTIVE.SCLK/IDLE.SCLK	Active Polarity
00,10	0
01,11	1

Clock Phase: The phase of the SCK signal is determined by the field CK.SDA in SBCx_COMMAND

Both clock polarity and clock phase together determine the SPI mode as described earlier.

Data signal (SDO) output format: The output format of the SDO signal is determined by ACTIVE.SDA and IDLE.SDA fields in SBCX_COMMAND register. Both these fields should be programmed with the same value.

Transmit mode: Transmit mode is enabled by setting TXEN bit in SBCX_COMMAND register.

Receive mode: Receive mode is enabled by setting RXEN bit in SBCX_COMMAND register.

Chip select: The SPI CS signal can be controlled either by software or by hardware based on the value programmed in CS_SW bit in SBCX_COMMAND register.

If CS_SW bit is set to 1, the SPI CS works in software mode and any value programmed in CS bit in SBCX_COMMAND register appears on SPI CS. In this mode, software is responsible to generate appropriate chip select polarity by writing correct values in CS bit. This mode can be used to do data transfers that require multiple packets within a single chip select assertion.

If CS_SW bit is set to 0, the SPI CS works in hardware mode. In this mode, CS bit in SBCX_COMMAND register works as polarity of the SPI_CS signal. The SPI_CS will be driven active on a per-packet basis. It will be driven inactive in between packets and when the SPI bus is idle. In this mode, CS_DELAY determines the number of cycles the SPI_CS signal stays inactive in between two packets. Setting CS_DELAY to 0 gives a delay of 2 SPI_SCK cycles in between two packets if receive mode is enabled. Any other number N written to this field will give a delay of N + 2 cycles if receive mode is enabled. If only transmit mode is enabled, setting CS_DELAY to N will give a delay of N cycles.

23.1.1.5 Status Information

There are certain bits of status information in SBCX_STATUS register that software can use to determine the status of the currently ongoing transfer.

BSY bit: This bit is set to 1 at the start of every transfer and is cleared when the transfer has finished.

RDY bit: This bit is set to 1 at the end of every transfer. It is cleared when software writes a 1 to this bit.

RXF_UNR bit: This bit indicates an under run in RX FIFO and is set to 1 whenever an error is detected during a receive operation.

TXF_OVF bit: This bit indicates an overflow in TX FIFO and is set to 1 whenever an error is detected during a transmit operation.

RXF_FULL: This bit indicates whether the RX FIFO is full or not and is set to 1 when RX FIFO is full. If this bit is set to 1 during a reception and BSY is 1, it indicates that the controller has received data in the RX FIFO and is waiting for the firmware to read these data before receiving any more data.

RXF_EMPTY: This bit indicates whether the RX FIFO is empty or not and is set to 1 when RX FIFO is empty.

TXF_FULL: This bit indicates whether the TX FIFO is full or not and is set to 1 when TX FIFO is full.

TXF_EMPTY: This bit indicates whether the TX FIFO is empty or not and is set to 1 when TX FIFO is empty. If this bit is set to 1 during a transmission and BSY is 1, it indicates that the controller has transmitted all data from the TX FIFO and is waiting for firmware/APB DMA to write more data before continuing with the transmission.

CUR_BLOCK_COUNT: This field contains the no. of packets that have been transferred (sent or received) during current transfer.

If both transmit and receive are enabled and either RXF_FULL or TXF_EMPTY is set, controller will pause both the transmission and reception regardless of whether RXF_FULL or TXF_EMPTY is set.

23.1.1.6 Clock Initialization and Control

The SPI controller is targeted to run at 50 MHz at its interface to external SPI devices. The internal SPI controller clock should then run at 4X of the external SPI interface clock. Hence, the internal SPI controller clock sources for SPI 1, SPI 2 and SPI 3 all need to run at 200 MHz in this case. This 4X ratio is required for the correct operation of the SPI controllers.

The CLK_OUT_ENB_H register in the CAR (Clock and Reset) contains the enable bits for the 3 instances of the SPI controller – SPI 1 (bit 9 - CLK_ENB_SBC1), SPI 2 (bit 12 - CLK_ENB_SBC2) and SPI 3 (bit 14 - CLK_ENB_SBC3).

In addition, the clock source and divider registers CLK_SOURCE_SBC1, CLK_SOURCE_SBC2 and CLK_SOURCE_SBC3 registers contain the 2 bit field SBCx_CLK_SRC to decide the PLL clock source while the field SBCx_CLK_DIVISOR determines the divide ratio.

23.1.1.7 Programming Limitations

In packed transfers, you must follow a 3 step sequence.

- Program the bit length and packet parameters
- Enable packed mode
- Enable DMA

If the SPI interface is running at slow speeds such as at 1 MHz then there is an additional delay up to 1us required between enabling Packed mode and enabling DMA.

- Programming of both the bit lengths and pack_size. When in packed mode, the controller should not depend on bit-length. Alternatively, the controller can rely on bit_length if the controller is not in packed mode.
- In packed mode the number of blocks transferred need to be aligned by boundary. i.e. for a packed 4 the number of packets should be multiple of 8 and for packed 8 it has to be multiple of 4.
- With SPI clock is 4 times faster than APB clock then there is a restriction on the bit-length. If the number of bits are <4, then the busy goes low as soon as the words are received on the SPI clock. But it doesn't wait till these are written into FIFO. And since APB clock is running slow, possibility of data lost if less number of bits are transferred.

- Busy bit does not go low if the transfer is terminated in the middle. Hence termination in the middle of a transfer is not recommended.
- DMA should be programmed after APB is programmed in APB-DMA transfers.
  - If more than 32 words need to be transferred, then wait for the FIFO to be full before you start the SPI controller to transmit
  - 32 or less words need to be transferred, then first fill up the FIFO before the SPI controller is started to transmit
- Full duplex transfer in GO mode - First word in RXFIFO is always 0x00. It is recommended to use the DMA mode instead.
- Programming of IDLE_SCK and ACTIVE_SCK. The requirement is to always program the same value in IDLE_SCK and ACTIVE_SCK. In that case we can just use one field instead of two.
- Similar is the case for IDLE_SDA and ACTIVE_SDA as for IDLE_SCK and ACTIVE_SCK. Program IDLE_SDA and ACTIVE_SDA with the same value.
- Implementation of WAIT for preventing writes to command register till the current transfer is finished.
- Use of master and slave buffers. As of now they don't do anything except storing the last word transmitted/received from/to TX/RX FIFO.
- SS_SETUP bit is reserved.
- If the SPI interface clock is run at very low frequencies (such as 1 MHz), it may be necessary to add delays in the programming sequence as specified in (1) above so that the programming can take effect.

23.1.2 SPI Controller as a Slave

The SPI controller can also be used as a slave. In this mode the controller receives the chip select and the clock from the external master. In addition, data input and outputs are swapped on the two data lines - MOSI and MISO. The table below provides additional information on the SPI signals in master and slave configurations:

Table 75. SPI Signals in Master and Slave Configurations

	Slave	Master
MOSI	Input	Output
MISO	Output	Input
SPI_CSx_N	Input	Output
SCK	Input	Output

SCK

The slave derives the clock from the external master. It is required that between two consecutive words (up to 32 bits), 1 cycle delay be provided for correct operation. This is mandatory for both the DMA and GO mode of operation. If the packed mode is selected (either one of 4, 8, 16 or 32 bits), a 1 cycle delay is required between two consecutive packets.

SS_N

The chip select will need to be asserted by the external master and will stay asserted during the entire duration of the operation. If the chip select is de-asserted during the middle of a transfer, the SPI controller will cease operation until the chip select is asserted. If the external master continues to provide clocks while the chip select stays de-asserted, it has no effect on the SPI controller. It is required that the chip select be asserted and the operation completed in full if such an event should occur.

MOSI and MISO

The data pins MOSI and MISO are now swapped while the SPI controller is operated as a slave. The SPI controller now drives out the data on MISO (instead of on MOSI when it is used as a master) and receives data on MOSI (as supposed to on MISO when it is used as a slave).

PIO and DMA Modes

For large transfers of data over the SPI interface, APB DMA could be used and the DMA mode should be selected in the SPI. This will enable the transfer of 2**16 packets (4, 8, 16 or 32 bits in each packet) over the SPI interface. The PIO mode can also be used to fill up the FIFO in the controller but may require higher latency to transfer such high data rates.

23.1.3 SPI Peripheral Interface Registers

23.1.3.1 Programming Sequence for Basic Data Transmission

After the module-initialization (reset and clock programming), program the corresponding bits of the following registers, preferably in the same sequence:

For Go Mode:

COMMAND2: Set the type of transfer i.e. Receive (Bit31), transmit (Bit30)

COMMAND1: Set the bit_length (bits [4:0]), number of words (bits [9:5]), mode (mode0, mode1, mode2 and mode3) as below

Table 76. Go Mode Command 1 for Different Modes

Mode	ACTIVE_SCLK	IDLE_SCLK	CK_SDA	Notes
Mode0	00	00	0	Clock is positive polarity and the data is latched on the positive edge of SCK
Mode1	00	00	1	Clock is positive polarity and the data is latched on the negative edge of SCK
Mode2	01	01	0	Clock is negative polarity and the data is latched on the negative edge of SCK
Mode3	01	01	1	Clock is negative polarity and the data is latched on the positive edge of SCK

TX_FIFO: Write data into TXFIFO

COMMAND1: Enable the Go Bit (bit 30)

For DMA mode:

COMMAND2: Set the type of transfer i.e. Receive (Bit31), transmit (Bit30)

COMMAND1: Set the bit_length (bits [4:0]), and mode as described above. Please note that the bit length to be set accordingly depending on the PACK_SIZE (bits [22:21] when packed mode(Bit 20)is set in the DMA_CTL register

```
PACK_SIZE       BIT_LENGTH
00              0x03
01              0x07
10              0x0f
11              0x1f
```

DMA_CTL: Set the DMA_BLOCK_LENGTH (bits [15:0]), RX_TRIG (bits [19:18]), TX_TRIG (bits [17:16]). For interrupts set IE_RXC (bit27), IE_TXC(bit26) to 1. For packed mode set PACK_SIZE (bits [22:21]) and PACKED (bit 20).

> **Note:** PACKED should be set after programming all the other parameters (except DMA_EN) in the COMMAND,COMMAND2 and DMA_CTL registers after setting all the parameters set DMA_EN(bit 31) to 1.

23.1.3.2 SLINK_COMMAND_0

Used for the setting of bit_length, number of words, and for selecting the transfer mode. Chip Select can also be selected to be in HW mode as SW mode with both active high and active low polarities for supporting devices with varying CS polarities.

Serial Link-2 Sub Block Command Register

Offset: 000h | Read/Write: R/W | Reset: 0b0000xx000000xx00xx00000000000000

Bit	Reset	Description
31	0x0	ENB: RD/WD access to Data Register would start the next transfer. (This allows continuous Receive via RD of Buffer and Automated Transmit per WD of Buffer Register) 0 = DISABLE 1 = ENABLE
30	0x0	GO: Program 1 after all the other bits in the COMMAND2 and COMMAND are programmed to start the transfer HW clears this bit automatically after the transfer is done Clearing of the bit by SW will stop the Shifter and latch the partial data into buffer. 0 = STOP 1 = GO
29	0x0	WAIT: 1 = Hold APB Cycle from writing another data into COMMAND register until RDY 0 = NOP. Use of this bit is deprecated. 0 = NOP 1 = WAIT
28	0x0	M_S: 1 = Master Mode (internal Clock) 0 = Slave Mode (external Clock) 0 = SLAVE 1 = MASTER
25:24	0x0	IDLE_SCLK: 11 = Pull High 10 = Pull Low 01 = Driven High 00 = Driven Low (default) 0 = DRIVE_LOW 1 = DRIVE_HIGH 2 = PULL_LOW 3 = PULL_HIGH
23	0x0	CS_POLARITY3: 1 = CS3 active high 0 = CS3 active low 0 = LOW 1 = HIGH
22	0x0	CS_POLARITY2: 1 = CS2 active high 0 = CS2 active low 0 = LOW 1 = HIGH

Bit	Reset	Description
21	0x0	CK_SDA: 1 = Rising Edge 0 = Falling Edge (default) 0 = FALLING_EDGE 1 = RISING_EDGE
20	0x0	CS_POLARITY1: 1 = CS1 active high 0 = CS1 active low 0 = LOW 1 = HIGH
17:16	0x0	IDLE_SDA: 11 = Pull High 10 = Pull Low 01 = Driven High 00 = Driven Low 0 = DRIVE_LOW 1 = DRIVE_HIGH 2 = PULL_LOW 3 = PULL_HIGH
13	0x0	CS_POLARITY: 1 = CS active high 0 = CS active low 0 = LOW 1 = HIGH
12	0x0	CS_VALUE: 1 = CS is high 0 = CS is low 0 = LOW 1 = HIGH
11	0x0	CS_SW: 1 = CS controlled by SW 0 = CS controlled by hardware 0 = HARD 1 = SOFT
10	0x0	BOTH_EN: 1 = both lines transmit/receive 0 = one line transmit and other receive 0 = DISABLE 1 = ENABLE
9:5	0x0	WORD_SIZE: 31 = Thirty Two words (Max)
4:0	0x0	BIT_LENGTH: 31 = Thirty Two bit Transfers (Max)

23.1.3.3 SLINK_COMMAND2_0

SLINK COMMAND2 register -- This is in general used to select the type of transfer (TX/RX/Both)

Serial Link-2 Sub Block Command2 Register

Offset: 004h | Read/Write: R/W | Reset: 0b000000xx0000000xxxx0000000x0xx00

Bit	Reset	Description

Bit	Reset	Description
31	0x0	RXEN: Receive enable 0 = DISABLE 1 = ENABLE
30	0x0	TXEN: Transmit enable 0 = DISABLE 1 = ENABLE
29	0x0	SPC0: 1 = bi directional mode 0= Normal Mode 0 = NORMAL 1 = BIDIR
28:26	0x0	WAIT_PACK_INT: number of cycles between two packs in the DMA. Use of this field is deprecated. Use INT_SIZE 8 = number of cycles between 2 packs (Max)
23:22	0x0	FIFO_REFILLS: Number of transfers the CS should stay low for word sizes more than 32. This will enable to do the transfer of word sizes > 32 without using APB-DMA 0x00 For word_sizes 1 to 32 0x01 For word_sizes 33 to 64 0x10 For word sizes 65 to 96 0x11 For word sizes 97 to 128 0 = REFILL0 1 = REFILL1 2 = REFILL2 3 = REFILL3
21:20	0x0	SS_SETUP: number of cycles CS should stay inactive between packets 4 = number of cycles in setup for chip select (Max)
19:18	0x0	SS_EN: 11 = chip select3 10 = chip select2 01 = chip select1 00 = chip select0(default) 0 = CS0 1 = CS1 2 = CS2 3 = CS3
17	0x0	CS_ACTIVE_BETWEEN: 1 = CS active between two packets 0 = CS inactive between two packets 0 = LOW 1 = HIGH
12:8	0x0	INT_SIZE: number of IDLE cycles between two packets 31 = thirty two cycles between 2 packets
7	0x0	MODFEN: 1 = Enable Modef 0 = Disable Modef (default) 0 = DISABLE 1 = ENABLE
6	0x0	BIDIROE: When set to 1 SPI uses only one data line (mosi/miso) for Tx and Rx depending on Master/Slave mode. This has effect only when SPC0 is set to 1 1 = ENABLE Enable Output buffer 0 = DISABLE Disable Output buffer (default)
4	0x0	SPIE: 1 = ENABLE Enable SPIE interrupt 0 = DISABLE Disable SPIE interrupt

Bit	Reset	Description
1	0x0	SSOE: 1 = Enable 0 = Disable (default) 0 = DISABLE 1 = ENABLE
0	0x0	LSBFE: 1 = FIRST Transmit/Receive LSB first 0 = LAST Transmit/Receive LSB last

23.1.3.4 SLINK_STATUS_0

SLINK STATUS register: Read this register to know the status of the transfer. Error bit is set whenever errors such as Underflow/Overflow are encountered.

Serial Link-2 Sub Block Status/Control Register

Offset: 008h | Read/Write: R/W | Reset: 0b00000000101000x00000000000000000

Bit	Reset	Description
31	0x0	BSY: 1 = BUSY Controller is Busy 0 = IDLE Controller is Free
30	0x0	RDY: 1 = READY controller is Ready for transfer 0 = NOT_READY controller is Busy. Write 1 to clear the flag
29	0x0	ERR: Will be set to 1 by HW when Errors such as Underflow/overflow occurs. Write 1 to clear the flag 0 = OK 1 = ERROR
28	0x0	SCLK: SCLK input signal State 0 = LOW 1 = HIGH
27	0x0	RX_FLUSH: Flush the RX FIFO 0 = NOP 1 = FLUSH
26	0x0	TX_FLUSH: Flush the TX FIFO 0 = NOP 1 = FLUSH
25	0x0	RX_OVF: RX FIFO Overflow 0 = OK 1 = ERROR
24	0x0	TX_UNF: TX FIFO Underflow 0 = OK 1 = ERROR
23	0x1	RX_EMPTY: RX FIFO Empty 0 = NOT_EMPTY 1 = EMPTY

Bit	Reset	Description
22	0x0	RX_FULL: RX FIFO Full 0 = NOT_FULL 1 = FULL
21	0x1	TX_EMPTY: TX FIFO Empty 0 = NOT_EMPTY 1 = EMPTY
20	0x0	TX_FULL: TX FIFO Full 0 = NOT_FULL 1 = FULL
19	0x0	TX_OVF: TX FIFO Overflow 0 = OK 1 = ERROR
18	0x0	RX_UNF: RX FIFO Underflow 0 = OK 1 = ERROR
16	0x0	MODF: Mode fault 0 = OK 1 = ERROR
9:5	0x0	WORD: In GO mode indicates number of words transferred (word count)
15:0	0x0	BLK_CNT: number of blocks transferred (BLOCK count) during DMA
4:0	0x0	COUNT: In Go mode indicates number of bits transferred (bit count)

23.1.3.5 SLINK_MAS_DATA_0

Serial Link-2 Sub Block Transmit Data Page Buffer Register

Offset: 010h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	MASTER_BUFFER: Tx/Rx Shift Pattern

23.1.3.6 SLINK_SLAVE_DATA_0

Serial Link-2 Sub Block Slave Data Page Buffer Register

Offset: 014h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	SLAVE_BUFFER: Tx/Rx Shift Pattern

23.1.3.7 SLINK_DMA_CTL_0

SLINK DMA Control Register: Used in the DMA transfers. Set packed mode transfers and the trigger levels in this register.

Serial Link-2 Sub Block DMA Control Register

Offset: 018h | Read/Write: R/W | Reset: 0b0xxx00xxx00000000000000000000000

Bit	Reset	Description
31	0x0	DMA_EN: 1 = ENABLE DMA mode is enabled, 0 = DISABLE DMA disabled
27	0x0	IE_RXC: Interrupt enable on receive completion. 1 = ENABLE Enable interrupt generation at the end of a receive transfer. 0 = DISABLE Disable interrupt generation for receive.
26	0x0	IE_TXC: Interrupt enable on transmit completion. 1 = ENABLE Enable interrupt generation at the end of a transmit transfer. 0 = DISABLE Disable interrupt generation for transmit.
22:21	0x0	PACK_SIZE: Specifies the packet size during the DMA mode 00 = 4 bits in a pack 01 = 8bits in a pack 10 = 16 in a pack 10 = 32 in a pack 0 = PACK4 1 = PACK8 2 = PACK16 3 = PACK32
20	0x0	PACKED: Packed mode enable bit. 1 = Packed mode is enabled. This is only valid if BIT_LENGTH in SBCX_COMMAND register is set to 3, 7, 15 or 31. When enabled, all 32-bits of data in the FIFO contain valid data packets of either 8-bit or 16-bit length. 0 = Packed mode is disabled. 0 = DISABLE 1 = ENABLE
19:18	0x0	RX_TRIG: Receive FIFO trigger level. 00 = 1 word. DMA trigger is asserted whenever there is at least 1 word in the RX FIFO. 01 = 4 word. DMA trigger is asserted when there are at least 4 words in the RX FIFO. 10 = 8 word. DMA trigger is asserted when there are at least 8 words in the RX FIFO. 11 = 16 word. DMA trigger is asserted when there are at least 16 words in the RX FIFO. 0 = TRIG1 1 = TRIG4 2 = TRIG8 3 = TRIG16
17:16	0x0	TX_TRIG: Transmit FIFO trigger level. 00 = 1 word. DMA trigger is asserted whenever there is at least 1 word in the TX FIFO. 01 = 4 word. DMA trigger is asserted when there are at least 4 words in the TX FIFO. 10 = 8 word. DMA trigger is asserted when there are at least 8 words in the TX FIFO. 11 = 16 word. DMA trigger is asserted when there are at least 16 words in the TX FIFO. 0 = TRIG1 1 = TRIG4 2 = TRIG8 3 = TRIG16
15:0	0x0	DMA_BLOCK_SIZE: N = N+1 packets number of packets should be aligned in the packed mode transfers. packed mode --> Number of packets 3 multiple of 8 7 multiple of 4 15 multiple of 2 31 from 0 to N.

23.1.3.8 SLINK_STATUS2_0

Serial Link-2 Status2 Register

SLINK Status2 Register: Contains the empty count for the transmit FIFO and the full count for receive FIFO

Offset: 01ch | Read/Write: R/W | Reset: 0b000000xxxxxxxxxx100000

Bit	Reset	Description
21:16	0x0	RX_FIFO_FULL_COUNT: Indicates the number of words in the receive FIFO
5:0	0x20	TX_FIFO_EMPTY_COUNT: Indicates the number of empty slots in the transmit FIFO

23.1.3.9 SLINK_TX_FIFO_0

Serial Link-2 Sub Block TX FIFO Buffer Register

Offset: 100h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000 | Default: 0000.0000

Bit	Reset	Description
31:0	0x0	TX_FIFO_REGISTER: Tx/Rx Shift Pattern

23.1.3.10 SLINK_RX_FIFO_0

Serial Link-2 Sub Block RX FIFO Buffer Register

Offset: 180h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000 | Default: 0000.0000

Bit	Reset	Description
31:0	0x0	RX_FIFO_REGISTER: Tx/Rx Shift Pattern

23.2 SPI Serial Flash Controller

The Serial Flash interface (SFLASH) Controller interfaces Tegra® 2 Processor to SPI-capable devices such as Flash memories and ADC/DAC devices. Tegra 2 supports only master mode of SPI operation on this interface.

This interface is specifically intended for serial flash and similar devices. For a general purpose SPI interface, use one of the four SPI controllers (also referred to as SLINK or SBC).

Features

- Independent RX and TX FIFOs
- FIFO depth of 4 x 32 bits
- Software controlled bit-length from 0 to 31, resulting in packet sizes of 1 to 32 bits
- Packed mode support for bit length of 7 (8-bit packet size) and 15 (16-bit packet size)
- CS can be controlled by software
- Receive compare mode where the controller listens for a particular pattern on the incoming data before receiving the data in the FIFO
- DMA support
- Simultaneous receive and transmit supported
- Maximum frequency of device clock is 52 MHz, so the maximum data rate is 52 Mbits/sec

Clocking

The SFLASH module requires two clock sources:

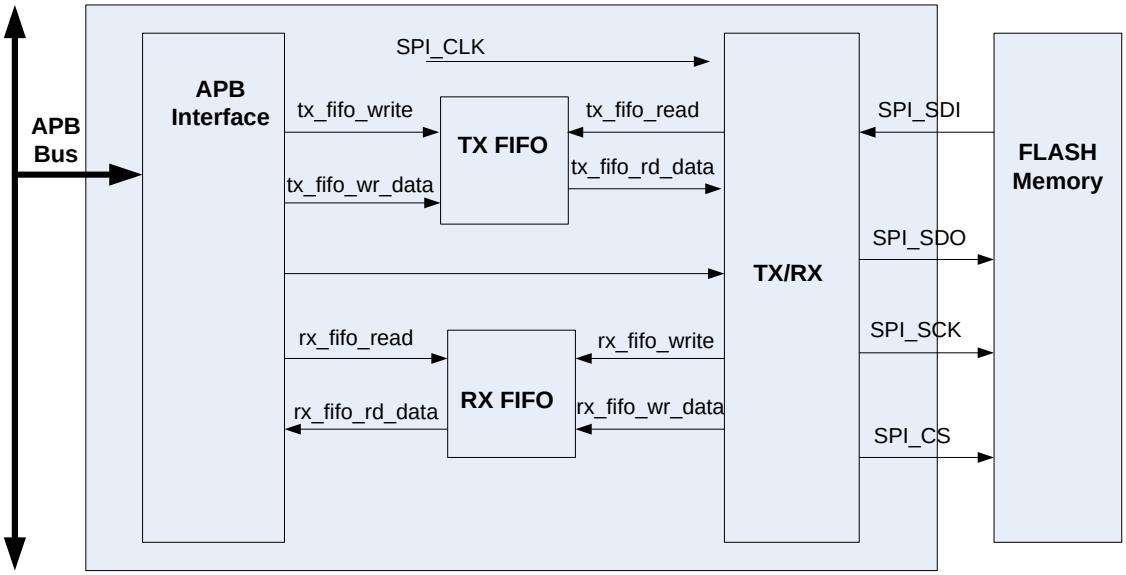
- **APB clock:** Max frequency is 150 MHz on this interface
- **Device/Interface clock:** Max frequency is 52 MHz on this interface

23.2.1 SPI Serial Flash Functionality

Tegra 2 Series SFLASH Controller works as a master on the SFLASH bus. It has independent transmit and receive FIFOs of 4 x 32 bits each. Software can program the controller to generate transactions of required packet length on the SFLASH bus, where a transaction is a sequence of packets in either direction.

It can use APB DMA 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 SFLASH bus. A block diagram for the SFLASH Controller is shown in the figure below.

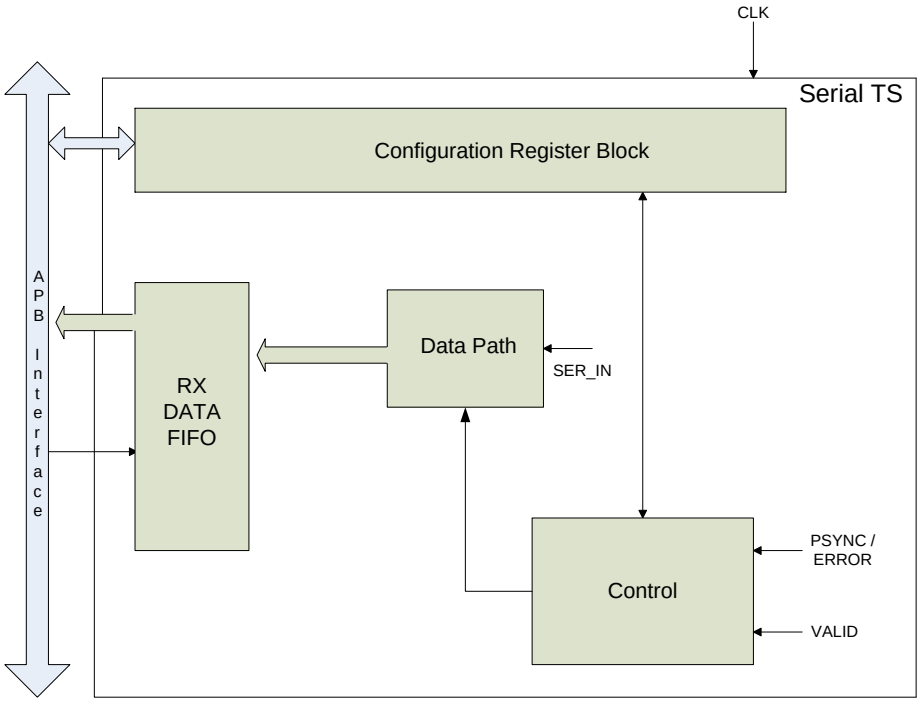
Figure 56. Serial FLASH Controller Block Diagram



23.2.2 DTV Functionality

The Transport Stream MDTV Controller works as a slave. It has Receive FIFO of 16 x 32 bits. Software programs the controller to generate transactions of required packet length, where a transaction is a sequence of packets in the receive direction. It uses APB DMA to read from the FIFOs as required. Software reads the Data and FEC from the FIFO and does the necessary operation. Figure below shows the TS with internal blocks.

Figure 57. Transport Stream with Internal Blocks



23.2.2.1 Signal Sanitization

The following internal signals are defined to provide flexibility to variations on the protocol:

Polarity normalized inputs:

- PSYNC = (TS_PSYNC ^ PSYNC_POL)
- VALID = (TS_VALID ^ VALID_ POL)
- ERROR = (TS_ERROR ^ ERROR_POL)

START: defined as one of the following (programming choice START_SELECT):

- Rising edge of VALID or
- Rising edge of PSYNC or
- Rising edge of (VALID & PSYNC)

START signal is internally used to delimit the beginning of the packet.

BODY_VALID: defined as one of the following (programming choice BODY_VALID_SELECT):

- ByteCounter = 0, where ByteCounter is a counter that gets loaded with BODY_SIZE value on START signal and decrements by 1 for each byte received. BODY_SIZE can be programmed to be in the inclusive range of 1-255 Bytes (default of 188).
- The interval between START and the falling edge of VALID.

FEC_VALID: asserted from the de-assertion of BODY_VALID and held high during FEC_SIZE Bytes.

- FEC_SIZE can be programmed to be in inclusive range of 0-127 Bytes (default 16).
- If FEC_SIZE = 0 this internal signal is not asserted at all.

PROTOCOL_SELECT: defined as one of the following (programming choice PROTOCOL_SELECT):

- 00b

  ERROR is tied to zero

  PSYNC is tied to zero
- 01b

  ERROR is tied to error/psync input port

  PSYNC is tied to zero
- 10b

  ERROR is tied to zero

  PSYNC is tied to error/psync input port
- 11b: Reserved

CLK_MODE: defines one of the following with the programmable option of DTV_MODE_0.clk_mode:

- 0 indicates a discontinuous clock source.
- 1 indicates a continuous clock source.

Discontinuous clock source is one where the clock toggles exactly 8*N times for a given packet size of N bytes. Any number of clocks greater than 8*N is regarded as continuous clock.

23.2.2.2 Protocol Handling

Serial Interface packet consists of a BODY section and a FEC section.

BODY: TS packet. BODY is validated by BODY_VALID. The first byte of the packet is expected to be a sync byte.

FEC: FEC_SIZE Bytes, FEC data (typically Reed-Solomon parity bytes). FEC is validated by FEC_VALID.

Packet capture stops after BODY and (if FEC_SIZE > 0) FEC are captured.

If START is re-asserted prior to completing a full packet, the packet will be considered an PACKET_UNDERRUN_ERROR. The sections below describe error conditions.

23.2.2.3 Error Conditions

A packet is considered an error packet in any of the following cases:

- **UPSTREAM_ERROR:** TS_ERROR is asserted during the first clock cycle of the packet and remains high during the capture of the whole packet.
- **BODY_UNDERRUN_ERROR:** If the number of bytes captured in Body (BODY SIZE) is < EXPECTED BODY_SIZE. Occurs only when DTV_CONTROL_0.body_valid_select is set to 1.
- **BODY_OVERRUN_ERROR:** If the number of bytes captured in BODY (BODY SIZE) is > EXPECTED BODY_SIZE. Occurs only when DTV_CONTROL_0.body_valid_select is set to 1.
- **PACKET_UNDERRUN_ERROR:** None of the above (1 through 3) occur and If total number of bytes captured (whole of the packet) is < EXPECTED (BODY_SIZE + FEC_SIZE)

The error conditions described above are checked in the order specified. The first one encountered is reported. A packet will produce at the most one error condition.

- Error packets are accounted on ERROR_COUNTER.
- Total packets are accounted on PACKET_COUNTER (regardless of whether they contain errors or not).
- Error packets interrupt the CPU with a status for each source of error. The byte stream stored in memory contains no explicit indication of this event.
- All packets are stored in memory regardless of its size.
- There is no specific H/W mechanism to signal S/W where the errors are located. S/W is expected to perform required checks to make sure the packet is correct.

Writing Error Packets into Memory

Error packets (including the case where they are signaled upfront i.e. UPSTREAM ERROR) are stored in memory with the captured length.

23.2.2.4 SERIAL TS Memory Format

Captured data is stored in memory according to the following format:

- BODY: BODY_SIZE Bytes out of which the first byte is considered a sync byte and
- FEC: FEC_SIZE bytes if FEC is present.

Description

- Start signal is generated as explained above with programmable option START_SELECT.
- VALID is asserted for Body and de-asserted for FEC, if at all it is taken into consideration with BODY_VALID_SELECT.
- Polarity of valid is decided with VALID_POLARITY.
- Sync byte is the first byte in body to indicate the start of a new packet. The value of sync byte is fixed as 0x47 to indicate the new packet is properly aligned.
- Either Error/Sync or both are tied to zero based on the programmed value in PROTOCOL_SELECT. Error/Sync polarity depends on ERROR_POLARITY and PSYNC_POLARITY respectively.

- Serially captured information on DATA is de-serialized (shift register) and stored in a FIFO.

23.2.2.5 Control

The control logic includes:

- Generation of start signal and the polarity for PSYNC / ERROR and VALID.
- Byte counter to count the number of bytes received in DATA and FEC.
- Assert the error bits on respective error conditions.
- End of packet generation based on the byte counter.

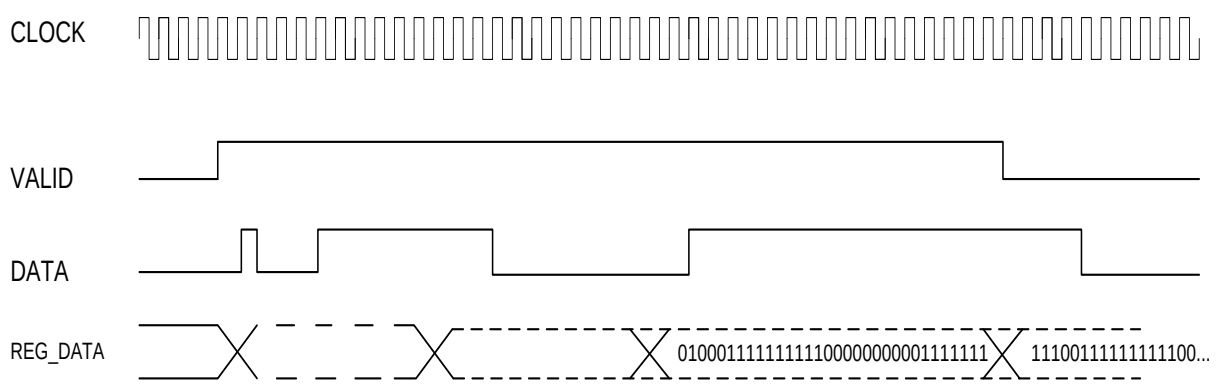
23.2.2.6 Data Path

Data path logic includes:

- De-serializing the serial information on DATA into 32 bit register to store into FIFO.
- To count the number of packets (non-error & error).

The de-serialization is as shown below:

Figure 58. De-serializing of Incoming Serial Data



23.2.2.7 Status Information

There are certain bits of status information in the registers that the software can use, to determine the status of the currently ongoing transfer.

RXF_OVF bit: This bit indicates an overflow in RX FIFO and is set to 1 whenever an error is detected during a receive operation.

RXF_FULL: This bit indicates whether the RX FIFO is full or not and is set to 1 when RX FIFO is full. If this bit is set to 1 during a reception, it indicates that the controller has received data in the RX FIFO and is waiting for the firmware to read these data before receiving any more data.

RXF_EMPTY: This bit indicates whether the RX FIFO is empty or not and is set to 1 when RX FIFO is empty.

INTERRUPT_STATUS: The respective bit is asserted high when one of the following occurs.

- Upstream error or
- Body Underrun error or
- Body Overrun error or
- Packet Underrun error.

The bit remains asserted until acted upon by the CPU to make it de-asserted.

PACKET_COUNTER.VALUE: This field contains the number of packets that have been transferred during current transfer irrespective of whether they are error or not.

PACKET_COUNTER.OVERFLOW: This bit is set high when the packet counter value overflows (passes from 0xFFFF to 0x0000).

ERROR_COUNTER.VALUE: This field contains the number of error packets. Increments when there is an upstream error or when one of the error conditions is asserted.

ERROR_COUNTER.OVERFLOW: This bit is set high when the error counter value overflows (passes from 0xFFFF to 0x0000).

23.2.3 SPI Serial Flash Programming Guidelines

There are two basic modes of operation: DMA_EN and GO mode. It is required that software sets up all the parameters in the SPI_COMMAND and SPI_DMA_CTL register before enabling transfers in any of these modes.

23.2.3.1 DMA_EN Mode

This mode is enabled by writing 1 to DMA_EN bit in SPI_DMA_CTL register. In this mode, SPI controller transmits or receives the no. of packets as indicated by the field DMA_BLOCK_SIZE in SPI_DMA_CTL register.

If PACKED bit is set and the BIT_LENGTH is set to 7, then all FIFO words contain 4 packets to transfer (transmit or receive). If DMA_BLOCK_SIZE is set to non-multiple of 4, then the last data word in the FIFO contains only the remaining packets and any extra bits after transferring required no. of packets will be ignored by the controller in case of transmit and will contain invalid data in case of receive. Packets will be transferred in little-endian format, with packet 0 being in byte 0 of the FIFO and packet 3 in byte 3 of the FIFO.

If PACKED bit is set and the BIT_LENGTH is set to 15, then all FIFO words contain 2 packets to transfer (transmit or receive). If DMA_BLOCK_SIZE is set to non-multiple of 2, then the last data word in the FIFO contains only the remaining packets and any extra bits after transferring required no. of packets will be ignored by the controller in case of transmit and will contain invalid data in case of receive. Packets will be transferred in little-endian format, with packet 0 being in bits [15-0] of the FIFO and packet 1 in bits [31-16] of the FIFO.

If BIT_LENGTH is set to N, each packet will consist of N + 1 bits. These bits will be transmitted/received in the TX_FIFO/RX_FIFO with the MSB in bit N and LSB in bit 0, following little-endian architecture. Any remaining bits in the FIFO will be ignored by the hardware. Maximum packet length is 32, which can be selected by setting BIT_LENGTH to 31. In this case, all data bits in FIFO contain valid packet data.

DMA request will be generated to APB_DMA in this mode depending on the setting of IE.TXC and IE.RXC. If transmit is enabled, setting IE.TXC to 00 will generate a TX DMA request whenever the TX FIFO has one word of space available (is not full). Setting IE.TXC to 01 will generate a TX DMA request whenever the TX FIFO has 4 words of space available (is empty). If receive is enabled, setting IE.RXC to 00 will generate a RX DMA request whenever the RX FIFO has one word of data available (is not empty). Setting IE.RXC to 01 will generate a RX DMA request whenever the RX FIFO has 4 words of data available (is full).

APB DMA requester number for SPI controller is 12.

23.2.3.2 GO Mode

This mode is enabled by writing 1 to GO bit in SPI_COMMAND register. In this mode, if transmit is enabled, then the SPI controller transmits the data in the TX FIFO until TX FIFO is empty. If receive is enabled, then the SPI controller receives one packet of data in the RX FIFO.

Packed mode is supported in transmit only. If PACKED bit is set and the BIT_LENGTH is set to 7, then all TX FIFO words contain 4 packets to transfer. If PACKED bit is set and the BIT_LENGTH is set to 15, then all TX FIFO words contain 2 packets to transfer.

23.2.3.3 Receive Compare Mode

Receive transfers have a special mode called receive compare mode which is enabled by setting RXCMP_EN bit in SPI_RX_CMP register. When this mode is enabled, all packets received on the SDI input are compared to a RXCMP_VAL value by applying a mask as specified in RXCMP_MASK. All packets received before this match occurs are ignored and only packets after this match occurs (including the matched packet) are saved to RX FIFO.

A 1 in any bit position in RXCMP_MASK will exclude that bit position in the received data from comparing against corresponding bit position in RXCMP_VAL. Only the bits that have 0 will be compared. This mode is only supported if the BIT_LENGTH is set to 7 (8-bits of packet length).

23.2.3.4 Interrupt Generation

SPI controller generates an interrupt to processor at the end of a transfer or when an error is detected if IE.TXC or IE.RXC bit in SPI_DMA_CTL register is enabled for transmit and receive modes of operation respectively. This functionality is common to both DMA_EN and GO modes.

If IE.TXC is enabled during transmit, an interrupt is generated whenever RDY or TX_UNR bit in SPI_STATUS register is set to 1.

If IE.RXC is enabled during receive, an interrupt is generated whenever RDY or RX_OVF bit in SPI_STATUS register is set to 1.

The interrupt can be cleared by clearing the source of the interrupt. If the interrupt is generated by assertion of RDY, then writing a 1 to RDY bit clears the interrupt. If the interrupt is generated by assertion of TX_UNR, then writing a 1 to TX_UNR bit clears the interrupt. If the interrupt is generated by assertion of RX_OVF, then writing a 1 to RX_OVF bit clears the interrupt.

SPI Controller interrupt source is bit SPI (bit 7) in secondary interrupt controller.

23.2.3.5 Packet Parameters

Clock signal (SFLASH_SCK) output format: The SCK clock signal output format is controlled by the fields ACTIVE.SCLK and IDLE.SCLK in SPI_COMMAND register. Both these fields should be programmed with the same value. These fields determine the active clock polarity as indicated below.

Table 77. Active Clock Polarity

ACTIVE.SCLK/IDLE.SCLK	Active Polarity
00, 10	0
01, 11	1

Clock Phase: The phase of the SCK signal is determined by the field CK.SDA in SPI_COMMAND register.

Both clock polarity and clock phase together determine the SPI mode as described earlier.

Data signal (SFLASH_DOUT) output format: The output format of the SDO signal is determined by ACTIVE.SDA and IDLE.SDA fields in SPI_COMMAND register. Both these fields should be programmed with the same value.

Transmit mode: Transmit mode is enabled by setting TXEN bit in SPI_COMMAND register.

Receive mode: Receive mode is enabled by setting RXEN bit in SPI_COMMAND register.

Chip select: The SFLASH_CSx_N signal can be controlled either by software or by hardware based on the value programmed in CS_SW bit in SPI_COMMAND register. If CS_SW bit is set to 1, the SFLASH_CSx_N works in software mode and any value programmed in CS bit in SPI_COMMAND register appears on SFLASH_CSx_N. In this mode, software is responsible to generate appropriate chip select polarity by writing correct values in CS bit. This mode can be used to do data transfers that require multiple packets within a single chip select assertion. If CS_SW bit is set to 0, the SFLASH_CSx_N works in hardware mode. In this mode, CS bit in SPI_COMMAND register works as polarity of the SPI_CS signal. The SFLASH_CSx_N will be driven active on a per-packet basis. It will be driven inactive in between packets and when the SPI bus is idle.

23.2.3.6 Status Information

There are certain bits of status information in SPI_STATUS register that software can use to determine the status of the currently ongoing transfer.

BSY bit: This bit is set to 1 at the start of every transfer and is cleared when the transfer has finished.

RDY bit: This bit is set to 1 at the end of every transfer. It is cleared when software writes a 1 to this bit.

TXF_UNR bit: This bit indicates an under run in TX FIFO and is set to 1 whenever an error is detected during a transmit operation.

RXF_OVF bit: This bit indicates an overflow in RX FIFO and is set to 1 whenever an error is detected during a receive operation.

RXF_FULL: This bit indicates whether the RX FIFO is full or not and is set to 1 when RX FIFO is full. If this bit is set to 1 during a reception and BSY is 1, it indicates that the controller has received data in the RX FIFO and is waiting for the firmware to read these data before receiving any more data.

RXF_EMPTY: This bit indicates whether the RX FIFO is empty or not and is set to 1 when RX FIFO is empty.

TXF_FULL: This bit indicates whether the TX FIFO is full or not and is set to 1 when TX FIFO is full.

TXF_EMPTY: This bit indicates whether the TX FIFO is empty or not and is set to 1 when TX FIFO is empty. If this bit is set to 1 during a transmission and BSY is 1, it indicates that the controller has transmitted all data from the TX FIFO and is waiting for firmware/APB DMA to write more data before continuing with the transmission.

CUR_BLOCK_COUNT: This field contains the no. of packets that have been transferred (sent or received) during current transfer. The bit SEL_TX_RX_N controls whether transmit or receive block count is read in this field.

If both transmit and receive are enabled and either RXF_FULL or TXF_EMPTY is set, controller will pause both the transmission and reception regardless of whether RXF_FULL or TXF_EMPTY is set.

23.2.3.7 Clock Initialization and Control

SPI controller clocks can be enabled or disabled by writing to SPI (bit 11) in CLK_OUT_ENB.H (0x6000:6014) register. Write a 1 to enable the clocks, and 0 to disable the clocks.

The clock frequency of SCK can be controlled by the CLK_SOURCE.SPI1 (0x6000:6114) register. The clock source is controlled by SPI1_CLK_SRC (bits [31:30]) of this register. The following are the valid values to select from.

Table 78. Clock Initialization and Control

SPI1_CLK_SRC	Selection	Description
00	PLLP_OUT0	Output 0 of PLLP
01	PLLC_OUT0	Output 0 of PLLC

SPI1_CLK_SRC	Selection	Description
10	PLLM_OUT0	Output 0 of PLLM
11	CLK_M	Clock M

This clock can be further divided by selecting appropriate divisor in SPI1_CLK _DIVISOR (bits [7:0]) in CLK_SOURCE.SPI1 (0x6000:6114) register.

Controller Reset

SPI controller can be reset by writing 1 to SPI1 (bit 11) in RST_DEVICES.H (0x6000:6008) register. Software needs to write a 0 to this bit to bring the SPI controller out of reset.

Pin-mux selection

SPI controller signals can be brought out to the following XM2 pins when using XM2_S pin-mux option.

Table 79. SPI Pin Mux Selection using XM2_S Option

SPI Signal	Description	Direction	XM2_S option
SFLASH_DIN	Serial Data In	Input	XM2_MUA[21]
SFLASH_DOUT	Serial Data Out	Output	XM2_MUA[20]
SFLASH_CS0_N	Chip select 0	Output	XM2_MCSA_N
SFLASH_CLK	Serial clock	Output	XM2_MCSB_N

If using CS1-CS3 pins, the following options can be used to bring them out.

Table 80. SPI Pin Mux Selection using CS1-CS3 Pins

SPI Signal	Description	Direction	Pin-mux option	Pin name
SFLASH__CS1_N	Chip select 1	Output	UAD	UART2_RXD
SFLASH__CS2_N	Chip select 2	Output	UAD	UART2_TXD
SFLASH__CS3_N	Chip select 3	Output	UCB	UC3_CTS

XM2_S option

To select this option, set XM2S_SEL (bits [31:30]) in PIN_MUX_CTL.C (0x7000:0088) to Alternate 2 (SPI) mode. Disable tristate on these pins by writing 0 to the field Z_XM2S (bit 31) in TRISTATE_REG_B (0x7000:0018).

Also, the GPIOs IO3_PJ.00, IO3_PJ.02, IO3_PB.00 and IO3_PB.01 should be set to special function mode.

CS1-CS3 signals

For CS1 and CS2, set UAD_SEL (bits [7:6]) in PIN_MUX_CTL.A (0x7000:0080) to Alternate 3 (SFLASH) mode. For CS3, set UCB_SEL (bits [19:18]) in PIN_MUX_CTL.B (0x7000:0084) to Alternate 0 (PWM0) mode.

Disable tristate on these pins by writing 0 to the field Z_UAD (bit 21) and Z_UCB (bit 23) in TRISTATE_REG_B (0x7000:0018).

Also, the GPIOs IO3_PC.02, IO3_PC.03 and IO3_PA.01 should be set to special function mode.

23.2.4 SPI Serial Flash Registers

23.2.4.1 SPI_COMMAND_0

SPI Command Register

Offset: 000h | Read/Write: R/W | Reset: 0b0x100xxxx0x00x00000010000100000

Bit	Reset	Description
30	0x0	GO: Default: 0 Go Mode enable bit. Software sets this bit to 1 to enable transmit or receive of packets without specifying the no. of packets. In receive mode, the controller receives one packet whenever software sets this bit. In transmit mode, controller transmits all data present in the TX FIFO until TX FIFO becomes empty. If packed mode is enabled, then all packets in the last word from the TX FIFO are transmitted before finishing the transfer. Software must set up all fields in SPI_COMMAND and SPI_DMA_CTL registers before setting this bit to 1. This bit clears to 0 by the hardware on the completion of the transfer. 0 = DISABLE 1 = ENABLE
28	0x1	M_S: Default: 1. Master/slave mode select. RO 1 = Controller is operating in master mode. 0 = Controller is operating in slave mode. This bit is read-only and fixed to 1. Only master mode is supported in this design. 0 = SLAVE 1 = MASTER
27:26	0x0	ACTIVE_SCLK: Active clock signal format. Controls the output enable of the SCK line when the controller is actively doing data transfers. 00: Drive low. 01: Drive high. 10: Pull low. 11: Pull high. Default: 00 0 = DRIVE_LOW 1 = DRIVE_HIGH 2 = PULL_LOW 3 = PULL_HIGH
21	0x0	CK_SDA: Clock phase. Controls how the data is transferred with respect to clock edge. 0 = Data is transferred on first clock edge after CS is driven low. 1 = Data is transferred on second clock edge after CS is driven low. 0 = FIRST_CLK_EDGE 1 = SECOND_CLK_EDGE
19:18	0x0	ACTIVE_SDA: Active Data signal format. Controls the output enable of the SCK line when the controller is actively doing data transfers. 00: Drive low. 01: Drive high. 10: Pull low. 11: Pull high. Default: 00. 0 = DRIVE_LOW 1 = DRIVE_HIGH 2 = PULL_LOW 3 = PULL_HIGH
16	0x0	CS_POL: CS signal Polarity. For both SW and HW CS modes, this bit works as the polarity of the CS signal Default:0 0 = ACTIVE_LOW 1 = ACTIVE_HIGH

Bit	Reset	Description
15	0x0	TXEN: Transmit enable. 1 = Transmit is enabled. Data is transmitted out from the TX FIFO to SDA output. 0 = Transmit is disabled. Default: 0 0 = DISABLE 1 = ENABLE
14	0x0	RXEN: Receive enable. 1 = Receive is enabled. Data is received on SDI line and placed in the RX FIFO. 0 = Receive is disabled. Default: 0 0 = DISABLE 1 = ENABLE
13	0x0	CS_VAL: CS signal value/polarity. If CS_SOFT is 1, then the value in CS_VAL is driven out on SPI_CS. If CS_SOFT is 0, then this bit works as the polarity of the CS signal and is driven to active state during packet transfers and inactive state in between packet transfers. 0 = LOW 1 = HIGH
12	0x0	CS_SOFT: Software control of SPI_CS signal 1 = SPI_CS is driven with the value in the CS bit. 0 = SPI_CS is driven to active during packet transfers by the hardware. Default: 0 0 = HW_CTL 1 = SW_CTL
11:9	0x2	CS_DELAY: Programmable delay between two packets if CS is used in hardware mode (CS_SOFT = 0). Default: 2.
8	0x0	CS3_EN: Enable for Chip select 3: 1 = cs3 is enabled. 0 = cs3 is disabled. (Default) 0 = DISABLE 1 = ENABLE
7	0x0	CS2_EN: Enable for Chip select 2: 1 = cs2 is enabled. 0 = cs2 is disabled. (Default) 0 = DISABLE 1 = ENABLE
6	0x0	CS1_EN: Enable for Chip select 1: 1 = cs1 is enabled. 0 = cs1 is disabled. (Default) 0 = DISABLE 1 = ENABLE
5	0x1	CS0_EN: Enable for Chip select 0: 1 = cs0 is enabled(Default). 0 = cs0 is disabled 0 = DISABLE 1 = ENABLE
4:0	0x0	BIT_LENGTH: Bit stream length. 0 = Single bit transfer. 1 = 2 bit transfer N = N + 1 bit transfer. 31 = 32 bit transfer (max) Default: 0

23.2.4.2 SPI_STATUS_0

SPI Status Register

Offset: 004h | Read/Write: R/W | Reset: 0b0000001010xxxxx00000000000000000

Bit	Reset	Description
31	0x0	BSY: Busy bit. Indicates that the controller is currently doing a data transfer. This bit is set at the start of every transfer and will be cleared at the end of every transfer. Default: 0 0 = NOT_BUSY 1 = BUSY

Bit	Reset	Description
30	0x0	RDY: Ready bit. This bit is set at the end of every transfer and an interrupt is also generated if the corresponding interrupt enable is set. Software writes a 1 to clear it. The interrupt is also cleared when this bit is cleared. Default: 0 0 = NOT_READY 1 = READY
29	0x0	RXF_FLUSH: RX FIFO Flush: WO. Software writes 1 to this bit to flush the RX FIFO. This bit will read 1 when the flush operation is in progress and will return to 0 when it is finished. Default: 0 0 = DISABLE 1 = ENABLE
28	0x0	TXF_FLUSH: TX FIFO Flush: WO. Software writes 1 to this bit to flush the TX FIFO. This bit will read 1 when the flush operation is in progress and will return to 0 when it is finished. Default: 0 0 = DISABLE 1 = ENABLE
27	0x0	RXF_UNR: RX FIFO Under run: RO. This bit is set to 1 whenever software tries to read from an empty RX FIFO. An interrupt is generated if the interrupt enable is set for receive operation (IE.RXC in SPI_DMA_CTL register). Software writes a 1 to clear this bit. Clearing this bit also clears the interrupt. Default: 0 0 = UNSET 1 = SET
26	0x0	TXF_OVF: TX FIFO Overflow: RO. This bit is set to 1 whenever software tries to write to a full TX FIFO. An interrupt is generated if the interrupt enable is set for transmit operation (IE.TXC in SPI_DMA_CTL register). Software writes a 1 to clear this bit. Clearing this bit also clears the interrupt. Default: 0 0 = UNSET 1 = SET
25	0x1	RXF_EMPTY: RX FIFO empty status: RO. Hardware sets this bit to 1 if RX FIFO is empty. Otherwise, this bit is set to 0. Default: 1. FIFO is empty at POR. 0 = NOT_EMPTY 1 = EMPTY
24	0x0	RXF_FULL: RX FIFO full status: RO. Hardware sets this bit to 1 if RX FIFO is full. Otherwise, this bit is set to 0. Default: 0. FIFO is empty at POR. 0 = NOT_FULL 1 = FULL
23	0x1	TXF_EMPTY: TX FIFO empty status: RO. Hardware sets this bit to 1 if TX FIFO is empty. Otherwise, this bit is set to 0. Default: 1. FIFO is empty at POR. 0 = NOT_EMPTY 1 = EMPTY
22	0x0	TXF_FULL: TX FIFO full status: RO. Hardware sets this bit to 1 if TX FIFO is full. Otherwise, this bit is set to 0. Default: 0. FIFO is empty at POR. 0 = NOT_FULL 1 = FULL
16	0x0	SEL_TX_RX_N: Selects whether the receive or transmit block count to be read in the field CUR_BLOCK_COUNT. 1: Transmit block count will be read in CUR_BLOCK_COUNT. 0: Receive block count will be read in CUR_BLOCK_COUNT. Default: 0. 0 = SEL_RX_CNT 1 = SEL_TX_CNT
15:0	0x0	CUR_BLOCK_COUNT: Selects whether the receive or transmit block count to be read in the field CUR_BLOCK_COUNT. 1: Transmit block count will be read in CUR_BLOCK_COUNT. 0: Receive block count will be read in CUR_BLOCK_COUNT. Default: 0.

23.2.4.3 SPI_RX_CMP_0

SPI Receive Compare Register

Offset: 008h | Read/Write: R/W | Reset: 0b00000000000000000

Bit	Reset	Description
16	0x0	RXCMP_EN: Enable Receive Compare mode. 1 = Enable receive compare mode. Data received on SDI signal is ignored until a compare match occurs, that is, if the mask on the data input by RXCMP_MASK matches the RXCMP_VAL. This is only valid if the BIT_LENGTH field in SPI_COMMAND register is set to 7 (8-bit packet length). 0 = Disable receive compare mode. All data received on SDI signal is placed in the RX FIFO. Default: 0 0 = DISABLE 1 = ENABLE
15:8	0x0	RXCMP_MASK: Mask on the receive data. This mask value is applied to the receive data before comparing it to RXCMP_VAL for a match. A 1 in any bit position in RXCMP_MASK will exclude that bit position in the received data from comparing against corresponding bit position in RXCMP_VAL. Only the bits that have 0 will be compared. Default: 0
7:0	0x0	RXCMP_VAL: Receive compare value. This value is compared to the received data after applying the mask in RXCMP_MASK. Default:0

23.2.4.4 SPI_DMA_CTL_0

SPI DMA Control Register

Offset: 00ch | Read/Write: R/W | Reset: 0b0xxx00xxxxx00000000000000000000000

Bit	Reset	Description
31	0x0	DMA_EN: Enable DMA mode transfer. Software writes a 1 to this bit to start a transfer in the DMA mode. All fields in the SPI_COMMAND and SPI_DMA_CTL register must be set before writing a 1 to this bit. This bit is cleared by the controller after all packets have been transferred as indicated by the DMA_BLOCK_SIZE field. Default: 0 0 = DISABLE 1 = ENABLE
27	0x0	IE_RXC: Interrupt enable on receive completion. 1 = Enable interrupt generation at the end of a receive transfer. 0 = Disable interrupt generation for receive. Default: 0 0 = DISABLE 1 = ENABLE
26	0x0	IE_TXC: Interrupt enable on transmit completion. 1 = Enable interrupt generation at the end of a transmit transfer. 0 = Disable interrupt generation for transmit. Default: 0 0 = DISABLE 1 = ENABLE
20	0x0	PACKED: Packed mode enable bit. 1 = Packed mode is enabled. This is only valid if BIT_LENGTH in SPI_COMMAND register is set to either 7 (8-bit transfer) or 15 (16-bit transfer). When enabled, all 32-bits of data in the FIFO contains valid data packets of either 8-bit or 16-bit length. 0 = Packed mode is disabled. Default: 0 0 = DISABLE 1 = ENABLE
19:18	0x0	RX_TRIG: Receive FIFO Trigger level 00: 1 word. DMA trigger is asserted whenever there is space for at least 1 word in the TX FIFO. 01: 4 words. DMA trigger is asserted when there is space for 4 words in the TX FIFO. 10: Reserved. 11: Reserved 0 = TRIG1 1 = TRIG4

Bit	Reset	Description
17:16	0x0	TX_TRIG: Transmit FIFO trigger level. 00: 1 word. DMA trigger is asserted whenever there is space for at least 1 word in the TX FIFO. 01: 4 words. DMA trigger is asserted when there is space for 4 words in the TX FIFO. 10: Reserved. 11: Reserved. Default: 00 0 = TRIG1 1 = TRIG4
15:0	0x0	DMA_BLOCK_SIZE: Size of data block to be transferred using DMA mode. This field specifies the size of the data block to be transferred through DMA mode. N: N + 1 Data packets. Default: 0

23.2.4.5 SPI_TX_FIFO_0

SPI TX FIFO Register

Offset: 010h | Read/Write: WO | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	SPI_TX_FIFO: TX FIFO

23.2.4.6 SPI_RX_FIFO_0

SPI RX FIFO Register

Offset: 020h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx | Default: 0000.0000

Bit	Reset	Description
31:0	X	SPI_RX_FIFO: RX FIFO

23.2.5 Digital Television (DTV)

Digital Television (DTV) is one of the three Digital Terrestrial Television Broadcasting (DTTB) systems in the world, along with DVB-T (Digital Video Broadcasting -Terrestrial) and ISDB-T (Integrated Services Digital Broadcasting-Terrestrial).

DVB-T, DVB-H, ISDB-T, DMB-T, DAB are all different mobile TV protocols. All these DTVs can individually be supported on different interfaces. One such interface is Transport Stream (TS) which can either be a serial or parallel communication system. Serial Transport Stream (Serial TS) is similar to the SPI protocol but is not directly compatible. The implementation of Serial TS Controller is described in this section. The controller is hooked up to the APB bus and acts as a slave.

Features

- SLAVE support for serial TS interface for different mobile TV protocols
- Programmable polarity for valid / packet sync and error signals
- Capture of Reed-Solomon data
- 0-20MHz frequency of operation
- Statistics counters
- DMA mastering capability

Hardware/Software Partitioning

- **Hardware** – Converts the serially incoming DATA / FEC to parallel and stores them to make them visible to software. Updates the status registers and generates interrupts when attention is required (for different error conditions).

- **Software** – Reads the DATA / FEC to perform necessary tasks, programs the control registers and handles the interrupts asserted by hardware.

Assumptions

- Supports only SLAVE mode of operation
- Programmable clock capture edge
- Minimum packet size to be 4bytes

23.2.6 DTV Registers

23.2.6.1 DTV_SPI_CONTROL_0

Offset: 040h | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
0	0x0	SELECT: 1 = DTV enabled 0 = SPI CTLR enabled 0 = DISABLED 1 = ENABLED

23.2.6.2 DTV_MODE_0

Mode selection register

Offset: 044h | Read/Write: R/W | Reset: 0b0000

Bit	Reset	Description
3:2	0x0	PROTOCOL_SELECT: For selecting the pin configuration for VD[1]. NONE : ERROR is tied to 0 PSYNC is tied to 0 ERROR : ERROR is tied to VD[1] PSYNC is tied to 0 PSYNC: ERROR is tied to 0 PSYNC is tied to VD[1] 0 = NONE 1 = ERROR 2 = PSYNC 3 = RESERVED
1	0x0	CLK_MODE: Determines if the input clock is continuous or discontinuous. 1 = CONTINUOUS 0 = DISCONTINUOUS 0 = DISCONTINUOUS 1 = CONTINUOUS
0	0x0	ENABLE: DTV protocol handling global enable. 1 = Enable 0 = Disable 0 = DISABLED 1 = ENABLED

23.2.6.3 DTV_CONTROL_0

Mode Control Register

Offset: 048h | Read/Write: R/W | Reset: 0b0010000010111100xxx00011x000x000

Bit	Reset	Description
30:24	0x10	FEC_SIZE: Stores the number of FEC bytes to capture, after the BODY has been captured.
23:16	0xbc	BODY_SIZE: // Stores the number of BODY bytes to capture including PSYNC.

Bit	Reset	Description
12:8	0x3	RX_FIFO_ATTN_LEVEL: Receive fifo trigger level 0x0 = 1 word Dma trigger is asserted when at least one word full in the fifo 0x1 = 2 word Dma trigger is asserted when at least 2 word full in the fifo

6	0x0	BODY_VALID_SELECT: Determines if VALID is used during BODY packet capture. IGNORE : VALID signal is ignored during the BODY capture. GATE : VALID gates the capture of BODY data. 7:7 rw STORE_UPSTREAM_ERROR_PKTS i=0x0 enum (DISCARD, STORE) // Determines if to store packets to memory that have been flagged as UPSTREAM_ERROR. DISCARD : Do not store packets in memory. STORE : Store UPSTREAM_ERROR packets in memory.
		0 = IGNORE
		1 = GATE

5:4	0x0	START_SELECT: Determines the START of the packet condition. PSYNC : PSYNC assertion rising edge. VALID : VALID assertion rising edge. BOTH : PSYNC && VALID asserted rising edge. 0 = RESERVED 1 = PSYNC 2 = VALID 3 = BOTH

2	0x0	ERROR_POLARITY: Indicates the polarity of ERROR, has effect only when MODE.ENABLE = ENABLED. LOW: Indicates the polarity of ERROR is active low. HIGH: Indicates the polarity of ERROR is active high. 3:3 rw CLK_POLARITY i=0x0 enum(HIGH, LOW) // Indicates the polarity of CLK, has effect only when MODE.ENABLE = ENABLED. LOW: Indicates the polarity of CLK is active low. HIGH: Indicates the polarity of CLK is active high.
		0 = HIGH
		1 = LOW

1	0x0	PSYNC_POLARITY: Indicates the polarity of PSYNC, has effect only when MODE.ENABLE = ENABLED. LOW: Indicates the polarity of PSYNC is active low. HIGH: Indicates the polarity of PSYNC is active high. 0 = HIGH 1 = LOW
0	0x0	VALID_POLARITY: Indicates the polarity of VALID, has effect only when MODE.ENABLE = ENABLED. LOW: Indicates the polarity of VALID is active low. HIGH: Indicates the polarity of VALID is active high. 0 = HIGH 1 = LOW

23.2.6.4 DTV_PACKET_COUNT_0

Packet Count Register

Offset: 04ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxx

Bit	Reset	Description
16	X	PACKET_COUNT_VALUE_OVERFLOW: Field is set to OVERFLOW when PACKET COUNT VALUE passes from 0xFFFF to 0x0000. Stays high until CPU writes a zero to this bit or resets it. 0 = NONE 1 = OVERFLOW
15:0	X	PACKET_COUNT_VALUE: Holds the current value of the received packet counter. This counter increments in the presence of a new packet, regardless of whether it is flagged error. The counter can be cleared by writing this register with 0's and can also be preloaded to any value by writing the preload value to the register.

23.2.6.5 DTV_ERROR_COUNT_0

Error Count Register

Offset: 050h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
16	X	ERROR_COUNT_VALUE_OVERFLOW: This field is set to OVERFLOW when ERROR COUNT VALUE passes from 0xFFFF to 0x0000. It stays high until the CPU writes a zero to this bit to reset it. 0 = NONE 1 = OVERFLOW
15:0	X	ERROR_COUNT_VALUE: Holds the current value of the error packet counter. This counter increments in the presence of a packet flagged as error or when protocol violation has occurred. The counter can be cleared by writing this register with 0's and can also be preloaded to any value by writing the preload value to the register.

23.2.6.6 DTV_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 will be reset. Interrupt status corresponding to the bits written with 0 will be left unchanged.

Offset: 054h | Read/Write: RO | Reset: 0b0000

Bit	Reset	Description
3	0x0	PACKET_UNDERRUN_ERROR_INT_STATUS: start condition detected prior to receiving a full packet 0 = Interrupt not detected 1 = Interrupt detected 0 = NOINTR 1 = INTR
2	0x0	BODY_OVERRUN_ERROR_INT_STATUS: Bit shows the status of the condition when the DTV input gets a body overrun error i.e. more bytes in body than specified 0 = Interrupt not detected 1 = Interrupt detected 0 = NOINTR 1 = INTR
1	0x0	BODY_UNDERRUN_ERROR_INT_STATUS: Bit shows the status of the condition when the DTV input gets a body underrun error 0 = Interrupt not detected 1 = Interrupt detected 0 = NOINTR 1 = INTR
0	0x0	UPSTREAM_ERROR_INT_STATUS: Bit shows the status of the condition when the DTV input gets an upstream error 0 = Interrupt not detected 1 = Interrupt detected 0 = NOINTR 1 = INTR

23.2.6.7 DTV_STATUS_0

DTV Status Register

Offset: 058h | Read/Write: R/W | Reset: 0b0xx

Bit	R/W	Reset	Description
2	RW	0x0	RXF_UNR: RX FIFO Underrun: RO. This bit is set to 1 whenever software tries to read from an empty RX FIFO. An interrupt is generated if the interrupt // enable is set for receive operation (IE.RXC in OWR_CTL register). Software writes a 1 to clear this bit. // Clearing this bit also clears the interrupt.
1	RO	X	RXF_EMPTY: RX FIFO empty status: RO. Hardware sets this bit to 1 if RX FIFO is empty Otherwise this bit is set to 0. 0 = NOT_EMPTY 1 = EMPTY

Bit	R/W	Reset	Description
0	RO	X	RXF_FULL: RX FIFO full status: RO. Hardware sets this bit to 1 if RX FIFO is full. Otherwise, this bit is set to 0. 0 = NOT_FULL 1 = FULL

23.2.6.8 DTV_RX_FIFO_0

DTV RX FIFO Register

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

Bit	Reset	Description
31:0	X	RD_DATA: RX FIFO

24.0 ONE WIRE BATTERY CONTROLLER

The One Wire Controller (OWR) implements a device communications bus system that provides low-speed data, signaling and power over a single signal. The OWR uses two wires for this--one for ground, and the other for power and data.

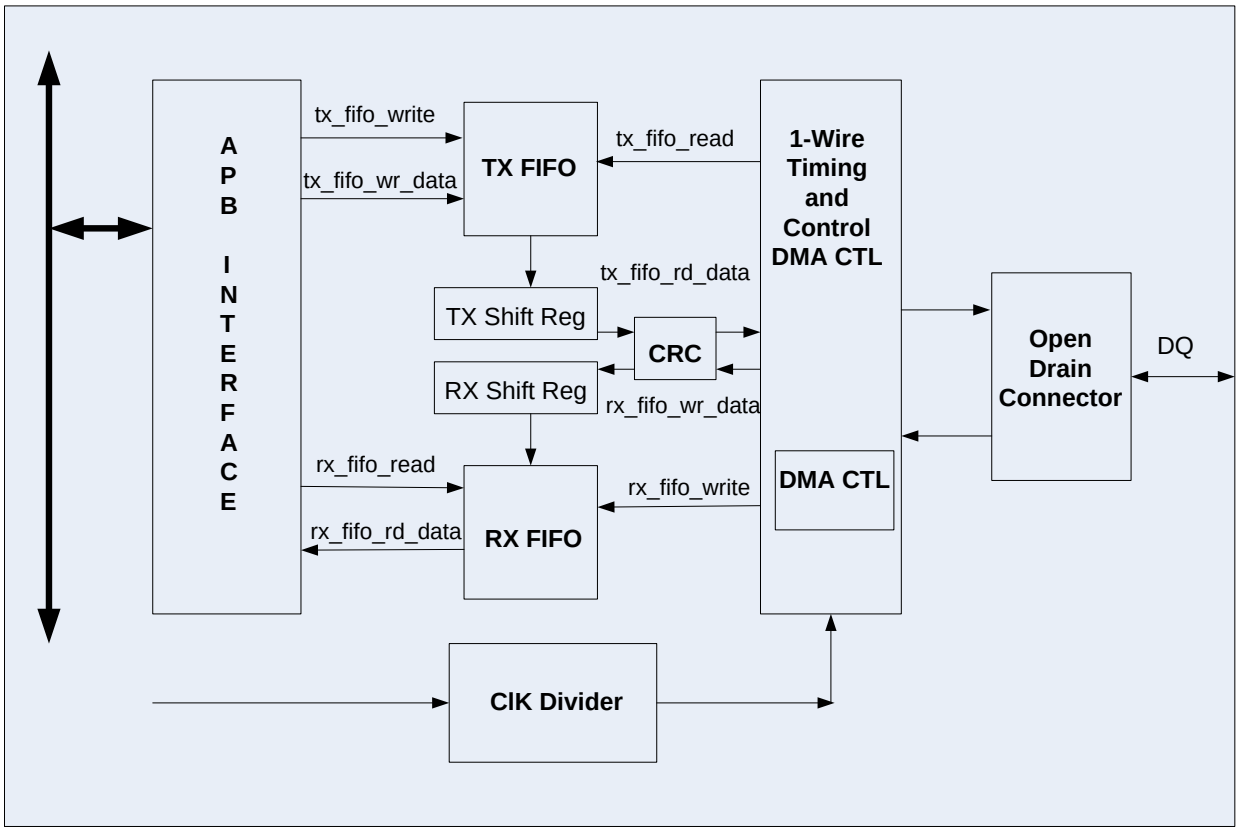
In Tegra® 2 Processor the one wire protocol is intended to communicate with battery controller chips.

Features

- Independent RX and TX FIFO's
- FIFO depth of 32 x 32 bits
- Hard-wired implementation of one wire protocol to eliminate need for external bridge chip
- Software programmable registers
- Software readable status registers
- Receive transmission with interrupt-based operation
- 1 MHz device clock required
- Supports de-glitch
- Supports Byte transfer or 1 Bit transfer
- Supports following commands: Read Rom, Skip Rom, Read Mem, Read Status, Read Data/Generate 8 bit CRC, Write Memory, Write Status
- Supports CRC 8/16 bit implementation
- Supports different battery devices, up to a memory size of 256KB in byte transfer

24.1 Functionality

Figure 59. OWR Block diagram



Specific command sequences as per the OWR protocol have to be followed for any data transaction over OWR interface.

The protocol involves first issuing a reset command where the controller looks for a presence from the device, and then issuing ROM commands before the EPROM is accessible, once the presence pulse is received.

The following ROM Commands are supported:

- Read ROM (33hex command)
- Skip ROM (CChex command)

When the read ROM command is issued, the controller receives a 64 bit ID from the device. This is an optional command, in case there is only a single slave present. One can directly issue a Skip Rom Command and issue memory commands to access EPROM.

EPROM accessible commands are:

- Read Memory (F0hex command)
- Read Status (AAhex command)
- Read Data/Generate 8 Bit CRC (C3hex command)
- Write Memory (0Fhex command)
- Write Status (55hex command)

The other commands can be used with 1-bit read/write. Contact your NVIDIA FAE for more details.

24.1.1 CRC Generation

To ensure data integrity, the controller has 8 bit and 16 bit CRC calculations. The equivalent polynomial function of this CRC is: $X^8 + X^5 + X^4 + 1$.

The CRC is calculated on the data being received from the device. During a Read Memory command the software has a provision to enable the CRC check bit where the CRC is calculated on the data received from the device (RD_MEM_CRC_REQ bit in Control Register).When this bit is enabled after the data is read from the EPROM, the device sends CRC to the controller .The controller then checks the received CRC against the calculated CRC and sets an crc_error flag in case of mismatch. The CRC can be 8 bit /16 bit depending on CRC_16BIT_EN bit in the Control Register. By programming the BY_PASS_CRC_ERR bit in the Control Register, the crc_error is ignored by the controller & continues with the command operation.

For other commands i.e. read data CRC, read status and write memory commands, the CRC checks are done by default as part of the protocol, unlike in read memory where it is optional.

Note: CRC 16bit is also supported in some battery devices.

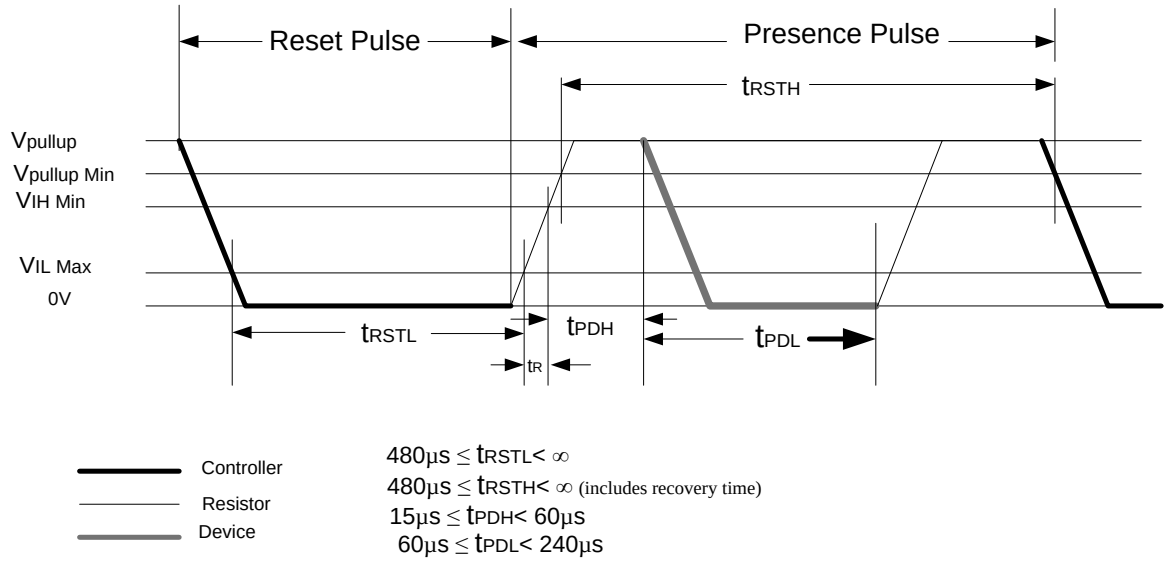
24.1.2 OWR Data Transfer Sequence

The OWR protocol consists of five types of data transfer:

- Reset & presence pulse
- Write 0
- Write 1
- Read Data
- Program Pulse

All data transfers (except presence pulse) are initiated by the controller. The communication should always begin with the initialization sequence, shown in the Figure 60 below.

Figure 60. Initialization Sequence



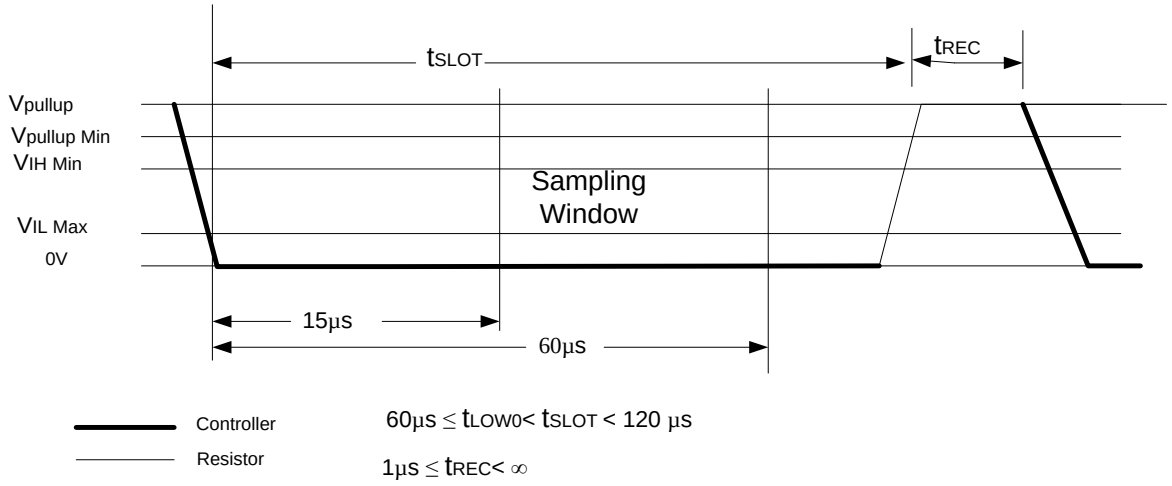
The reset pulse begins the initialization sequence. It is initiated when the GO bit in the control register (Control[GO]) is set.

24.1.2.1 Read/Write Time Slots

Write 0 Timing Slot

During a write memory command, for writing a bit 0, the data can be sampled and written into the scratch pad any time between t_{LOW0} and t_{SLOT} duration as shown in Figure 61 below

Figure 61. Write 0 timing



Write 1/Read Data Timing Slot

The Write 1 and Read timing slots are identical. The data bus is pulled low for t_{LOW1} time. The device can sample the data in sampling window (Figure 62). Similarly during a read slot the controller can read the data from the device in sampling window.(Figure 63)

Figure 62. Write 1 Timing

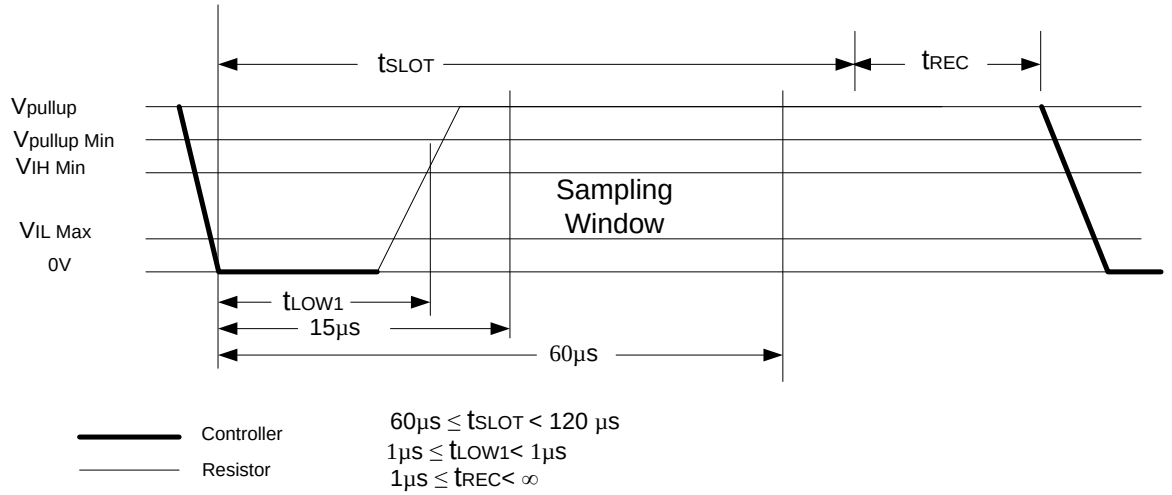
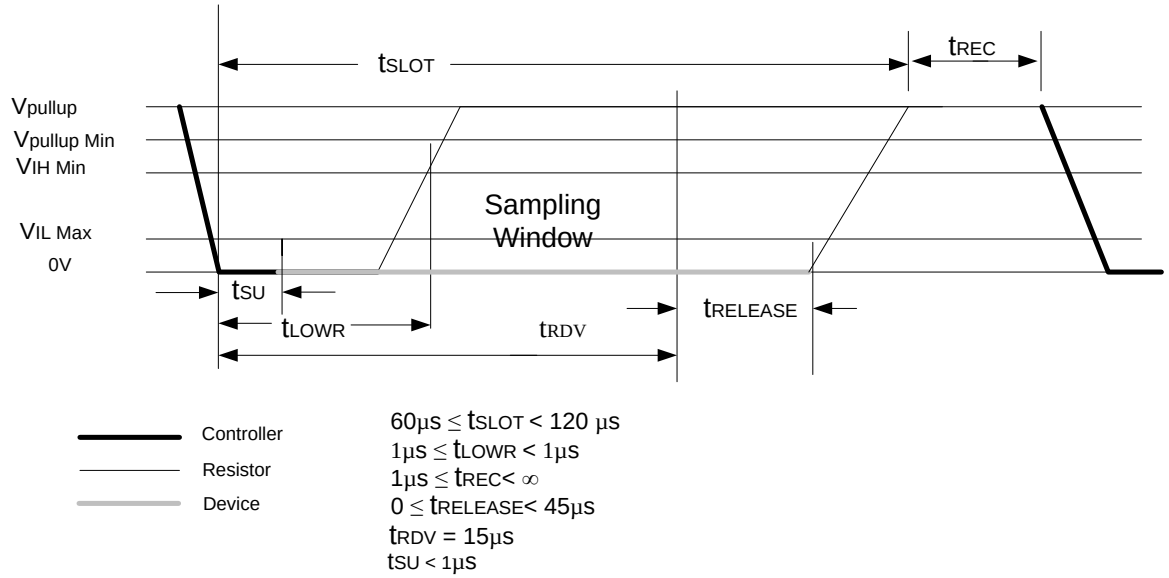


Figure 63. Read Timing

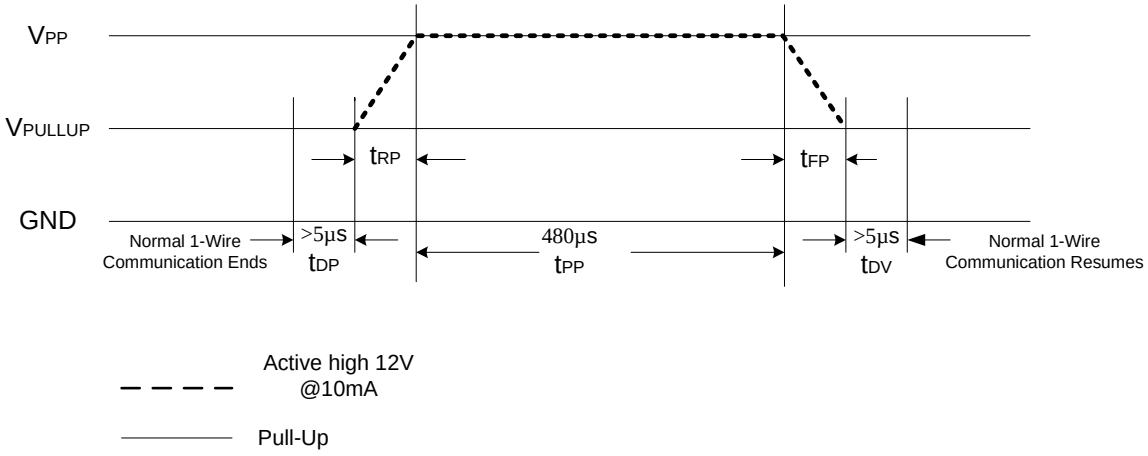


24.1.2.2 Programming Pulse

During a write memory command, data is first copied to the scratch pad. To shift the data from the scratch pad to the EPROM, a program pulse of 12 volts is applied to the data line after the controller has confirmed that the CRC is correct.

- In byte mode transfer the controller controls the generation of programming pulse.
- In single bit Write mode programming pulse is generated using GPIO's
- In Programming pulse window OWR bus is controlled by 12V voltage providing minimum of 10mA of current.

Figure 64. Programming Timing Pulse

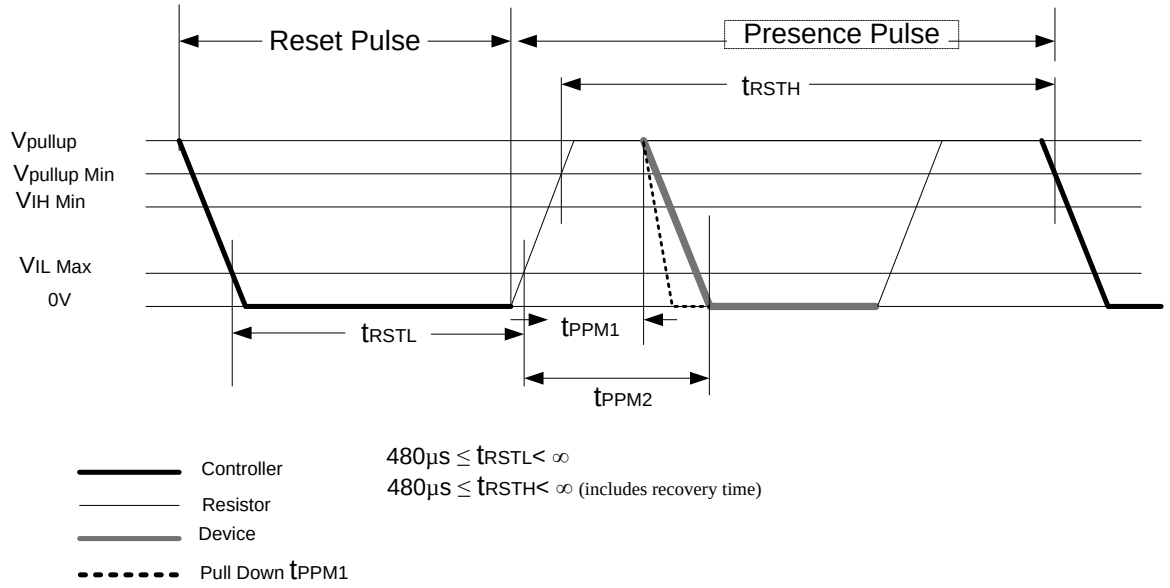


24.1.2.3 Presence-Pulse Masking (PPM)

Presence-Pulse Masking is used when battery devices are propagated through networks and get reflected. Glitches may occur due to reflections in network which may cause slave devices to lose synchronization.

When Presence-Pulse Masking is enabled, the controller pulls the OWR bus to low at t_{PPM1} till t_{PPM2}. At t_{PPM2} battery device further pulls down to low.

Figure 65. Presence-Pulse Masking



24.1.3 Interrupts

Interrupts are used to indicate the status of the design at different stages, so that the next data transfer can be determined.

Done signals indicate the data transfer has been done without any interruptions.

Error signals indicate an error has occurred and data transfer should start again. Error can be CRC err, presence error, memory write error etc.

Overflow/underrun indicates the FIFO status.

Data request indicates FIFO's are ready to accept or send the data.

All the Interrupts can be controlled by Interrupt MASK enables and Interrupt SET register.

24.1.4 Error handling

If OWR controller receives an error like CRC error, presence error, memory write error, or an error command; a reset pulse must be issued and the entire sequence must be repeated.

The OWR controller can bypass these errors by enabling register bits in the control register.

CTL.BYPASS_CRC_ERR: Transfer doesn't stops on CRC error

CTL.BYPASS_DGLITCH: Stops the execution of deglitch logic

CTL.PRESENCE_PULSE_MASKING: Transfer doesn't stops on presence error.

24.1.5 Clock Control

OWR controller clocking has following clock source options:

Option 0: pllP_out0 = 432MHZ (Fixed 432 MHz directly from PLLP)

Option 1: pllC_out0 = Up to 600MHz, Programmable clock output directly from PLLC.

Option 2: pllM_out0 = Up to 400MHz, Programmable clock output directly from PLLM.

Option 3: dbg_oscout = External clock source determined by the customer. 12 MHz, 13 MHz, 19.2 MHz, or 26 MHz are supported frequencies.

An 8-bit divider is used to generate a 1 MHz clock

The clock divider generates a 1 MHz clock that is used as a time reference by the state machine. Transitions between the states of the state machine as well as actions triggered at precise time deadlines are expressed using the 1-MHz clock. The state machine performs all required actions to dialog with the external device.

24.1.6 Deglitch/Debounce

To overcome the glitches on the input dq line, deglitch logic is used to make sure the data is stable for at least one microsecond. If a glitch occurs, an interrupt is generated and the transfer should start again. There is a bit to bypass deglitch logic. If that bit is set, deglitch is bypassed and if glitches occur they are not reported.

When to sample the data is determined by programming the control reg presence_sample_clk for reset initialization and read_data_sample_clk for read time slots.

There is a limitation to the value programmed in Presence/read_data sample clk: the hardware requires 6 clock cycles to implement deglitch generation logic (Synchronizer output requires 2 clks and deglitch 1 clk. If there is a glitch on dq line, HW samples the data on next clk edge, 2 clks later, to store the sampled data 1 clk pulse).

Presence sample clk = Tpdl - 6 clks

Rd_data_sample_clk = Trdv - 6 clks

24.1.7 Read/Write 1 Bit

The controller also supports sending or receiving data bit by bit. In bit by bit transfer, the software:

- calculates all the required calculations like when to send the wr0/wr1/rd
- calculates the CRC for received data
- check CRC and
- preserves the received data

The hardware does not preserve any data in registers, except the received 1 bit data, in RD_SAMPLE_DATA register.

The controller can only send the Time slots (Wr1/Wr0/Rd) and store the received sampled data in a reg. Time slots are controlled by the value programmed in the timing registers.

In case of writes the program pulse generation can be done using GPIO pins.

24.1.7.1 Programming Sequence

OWR communication always begins with reset initialization. To start reset initialization, program the CTRL.GO Bit in the control register. Program all the timing registers before the GO Bit. If device presence is detected, program the rom command and then program Memory Command.

Programming Rom Cmd

Transmit LSB first to send ReadRom Command (33h, 0011-0011).

Time Slots

- LSB of ReadRom command is 1'b1. Program CTRL.WR1_BIT in control register. After controller generates write-1 time slot, wait for the write-1 time slot to be completed by looking into INTR_STATUS.BIT_TRANSFER_DONE.
- After this is complete, program the next bit (in this case it's again 1'b1). Program CTRL.WR1_BIT in control register. After Controller generates write-1 time slot, wait for the write-1 time slot to be completed by looking into INTR_STATUS.BIT_TRANSFER_DONE
- Program the next bit then (in this case it is 1'b0). Program CTRL.WR0_BIT in control register. After controller generates write-0 time slot, wait for the write-0 time slot to be completed by looking into INTR_STATUS.BIT_TRANSFER_DONE

Repeat this for the remaining bits.

- After the ReadRom Command is sent, read 64 bits rom data. To receive 64 bits, 64 read time slots have to be sent. To generate Read time slot, program CTRL.RD_BIT in control register. After the controller generates read time slot, the read bit is stored in READ_SAMPLED_BIT. Sampled Bit should always be read after the bit transfer is done.
- Next, program the CTRL.RD_BIT, to read 2nd bit. Read time slot is generated by the controller. Wait for done and read the sampled bit.

Repeat the sequence for remaining bits.

After receiving all 64 bits, Memory data has to be read.

Reading EPROM data using ReadMem Command

- To read Memory data, first send memory cmd followed by memory address. After memory cmd and address is sent, verify the data sent. This can be done by checking CRC of cmd and address sent. To receive CRC, send 8 read-slots, read the CRC data and check with computed value if it matches. Read the EPROM data from memory or else start with initialization procedure.
- To read data from EPROM, send read time slots. The number of read time slots should be memory address – eprom endoffset. After reading EPROM data, read CRC and check the received data if CRC is required. Normally CRC is embedded within the EPROM data. To receive 8 bit CRC data send 8 read time slots.

24.2 Programming Guidelines

24.2.1 Programming Registers

All the registers should be programmed and CTL.GO bit should be programmed to start the controller.

Registers to Program Prior to the Control Register

- **All the timing registers**: Program the timing values according to battery device data sheet
- **Command reg**: Program the device Rom Cmd, Mem Cmd, EPROM starting address
- **EPROM size offset**: Program the number of bytes to transfer
- **Interrupt mask enables**: Program the respective interrupt mask enables to enable the interrupts to CPU/COP
- **Control Reg**: Programming this register depends on the type of execution required.

Program the G0 bit only after programming all of the above.

Registers to Monitor

- **Interrupt Status & Source**: These are used to monitor the design status (like done, error etc).
- **CRC**: Stores the calculated CRC 8/16 bit value

- **Byte Count**: Indicates the number of bytes transferred/received
- **Read_ROM0/ROM1**: Stores the received Rom Family code, Serial Number, CRC.

FIFO Registers

- Tx/Rx FIFO: Data to transmit/Receive data.

24.2.2 Byte Data Transfer Sequence

Reset Initialization → Rom Cmd → Mem Cmd → Reset Initialization → Rom Cmd → Mem Cmd etc

In case of an error, the entire sequence should start again from the beginning:

Reset Initialization → Error → Reset Initialization →Rom Cmd → Error → Reset Initialization → Rom Cmd → Mem Cmd etc.

24.3 OWR Registers

24.3.1 OWR_CONTROL_0

This register is the main control register. It should be configured last after configuring all other settings.

OWR control Register

Offset: 000h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	RD_BIT: if 0 no transfer is done. If 1 read time slot is executed. This bit is a write only. Read to this register will return 0 0 = NO_TRANSFER 1 = TRANSFER_READ_SLOT
30	0x0	WR0_BIT: if 0 no transfer is done. If 1 write zero time slot is executed. This bit is a write only. Read to this register will return 0 0 = NO_TRANSFER 1 = TRANSFER_ZERO
29	0x0	WR1_BIT: if 0 no transfer is done. If 1 write one time slot is executed. This bit is a write only. Read to this register will return 0 0 = NO_TRANSFER 1 = TRANSFER_ONE
28	0x0	BY_PASS_CRC_ERR: this bit is set to 1, if transfer needs to continue on crc err else on err transfer stops, and again transfer starts on setting rpp reset(go bit) 0 = STOP_TRANSFER_ON_CRC_ERR 1 = CONTINUE_TRANSFER_ON_CRC_ERR
27	0x0	BY_PASS_DGLITCH: This bit is used to bypass the deglitch logic, If 1, just takes the sync output If 0, looks for any glitch in the sample window for at least 1us, Deglitch requires a minimum of 6 clks(2 for sync, 2 for deglitch, if glitch, checks for 2 more clks, still glitch exists, err interrupt is asserted and data transfer should start from first) 0 = START_DGLITCH 1 = NO_DGLITCH
26:23	0x0	RD_DATA_SAMPLE_CLK: read data sample window, master samples the data_inwhich should be less than or equal to (tlow1 - 6) clks 6 clks are used for Deglitch, if Deglitch bypassed 3 clks should be enough
22:15	0x0	PRESENCE_SAMPLE_CLK: presence pulse sample clk, master samples the data_in which should be less than or equal to (tpdl - 6) clks 6 clks are used for dglitch, if Deglitch bypassed 3 clks should be enough
14	0x0	RD_MEM_CRC_REQ: This bit is set to 1, if crc is required for read memory cmd at end of memory 0 = NO_CRC_READ

Bit	Reset	Description
		1 = CRC_READ
13:9	0x0	RX_FIFO_ATTEN_LEVEL: Receive fifo attention level 000 = 1 word, fifo req is asserted when least one word full in the fifo 001 = 2 word, fifo req is asserted when least 2 words full in the fifo etc.....
8:4	0x0	TX_FIFO_ATTEN_LEVEL: Transmit fifo attention level 000 = 1 word, fifo req is asserted when least one word empty in the fifo 001 = 2 word, fifo req is asserted when least 2 words empty in the fifo etc.....
3	0x0	CRC_16BIT_EN: if set to 1 16bit crc is executed if set to 0 8bit crc is executed 0 = CRC_8BIT_EN 1 = CRC_16BIT_EN
2	0x0	DATA_TRANSFER_MODE: if set to 1 data transfer is done bit by bit if set to 0 data transfer is done through byte 0 = BYTE_TRANSFER_MODE 1 = BIT_TRANSFER_MODE
1	0x0	PRESENCE_PULSE_MASKING: when set, dq is driven to low by master before the slave does clearing this bit disables the ppm 0 = NO_PPM 1 = START_PPM
0	0x0	GO: Generate Reset Presence Pulse This bit is a write only read to this register will return 0this bit should be programed after all the registers are programed 0 = NO_PRESENCE_PULSE 1 = START_PRESENCE_PULSE

24.3.2 OWR_COMMAND_0

Rom Cmd, Mem Cmd and Mem Addr should be program at the same time.

OWR Command Register

Offset: 004h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	MEM_ADDR: Eprom Starting Address[15:0] to write/read data into Eprom
15:8	0x0	MEM_CMD: one wire MEM commands
7:0	0x0	ROM_CMD: one wire ROM commands

24.3.3 OWR_EPROM_0

Indicates the number of bytes to transfer, supports up to 256K EPROM size. MEM_ADDR and MEMORY_BYTES_TRANSFER/STATUS_BYTES_TRANSFER should always be in sync as in if EPROM end offset address is 7f, and we want to read data from 4th location address of EPROM.

Programming should be in the following way:

MEM_ADDR = 0x4;

MEMORY_BYTES_TRANSFER = 0x7B; (0x4 - 0x7f) similarly for status bytes.

OWR Eprom Size offset

Offset: 008h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	STATUS_BYTES_TRANSFER: Num of Eprom Status bytes to transfer Mem_Addr - Status bytes end address

Bit	Reset	Description
15:0	0x0	MEMORY_BYTES_TRANSFER: Num of Eprom memory bytes to transfer, Mem_Addr - Eprom end address

24.3.4 OWR_WR_RD_TCTL_0

Controls the one wire dq output(Pullup or pullDown) to write and read time slots timings are in micro seconds. Note : <= means less than or equal to

OWR Write Read Timing Control Register

Offset: 00ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
29:28	0x0	TSU: Read Data Setup, Tsu = N owr clks, Range = tsu < 1
27:22	0x0	TRELEASE: Release one wire Time, Trelease = N owr clks, Range = 0 <= trelease < 45
21:18	0x0	TRDV: Read data valid time, Trdv = N+1 owr clks, Range = Exactly 15
17:11	0x0	TLOW0: Write Zero time Low, Tlow0 = N+1 owr clks, Range = 60 <= tlow0 < tslot < 120
10:7	0x0	TLOW1: Write one time Low, or TLOWR both are same Tlow1 = N+1 owr clks, Range = 1 <= tlow1 < 15 TlowR = N+1 owr clks, Range = 1 <= tlowR < 15
6:0	0x0	TSLOT: Active time slot for write or read data, Tslot = N+1 owr clks, Range = 60 <= tslot < 120

24.3.5 OWR_RST_PRESENCE_TCTL_0

Controls the one wire dq output(Pullup or pullDown) to reset initialization time slots timings are in micro seconds.

Note : <= means less than or equal to

OWR Reset Presence Timing Control Register

Offset: 010h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:24	0x0	TPDL: PRESENCE_DETECT_LOW Tpdl = N owr clks, Range = 60 <= tpdl < 240
23:18	0x0	TPDH: PRESENCE_DETECT_HIGH Tpdh = N+1 owr clks, Range = 15 <= tpdh < 60
17:9	0x0	TRSTL: RESET_TIME_LOW Trstl = N+1 owr clks, Range = 480 <= trstl < infinity
8:0	0x0	TRSTH: RESET_TIME_HIGH, Trsth = N+1 owr clks, Range = 480 <= trsth < infinity

24.3.6 OWR_PPM_CORRECTION_TCTL_0

Drives the dq line to low before slave does

OWR Presence Pulse Masking Timing Control

Offset: 014h | Read/Write: R/W | Reset: 0b0000000000000000

Bit	Reset	Description
15:6	0x0	TPPM2: PRESENCE PULSE MASK STOP
5:0	0x0	TPPM1: PRESENCE PULSE MASK START

24.3.7 OWR_PROG_PULSE_TCTL_0

Controls the 12v programming pulse used to write data to eprom timings are in micro seconds

OWR Program Pulse Timing Control Register

Offset: 018h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	TPP: Program Pulse Width Tpp = N owr clks Range = 480 to 5000
15:12	0x0	TFP: Program Voltage Fall Time Tfp = N owr clks Range = 0.5 to 5
11:8	0x0	TRP: Program Voltage Rise Time Trp = N owr clks Range = 0.5 to 5
7:4	0x0	TDV: Delay to verify Tdv = N owr clks, Range = > 5
3:0	0x0	TPD: Delay to program Tpd = N+1 owr clks, Range = > 5

24.3.8 OWR_READ_ROM0_0

Read ROM Cmd Reads 8-bit family code, 48 bit serial Num and 8 bit crc. Total of 64 bits, Register can have a max of 32 bits. Read Rom Cmd data is split into 2 registers 1. READ_ROM0 2. READ_ROM1

OWR Read ROM DATA0 Register

Offset: 01ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:8	X	SERIAL_NUM0: Reads the first 24 bits of rom serial number
7:0	X	FAMILY_CODE: Reads the 8 bit family code of ROM

24.3.9 OWR_READ_ROM1_0

OWR Read ROM DATA1 Register

Offset: 020h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:24	X	CRC_BYTE: Reads the 8 bit CRC code of ROM
23:0	X	SERIAL_NUM1: Reads the next 24 bits of rom serial number

24.3.10 OWR_INTR_MASK_0

This register stores the masks for the interrupts. If a bit is set 1 and the corresponding interrupt condition is satisfied, interrupt line to system interrupt controller is asserted.

OWR INTR Mask Register

Offset: 024h | Read/Write: R/W | Reset: 0b00000000000000

Bit	Reset	Description
13	0x0	BIT_TRANSFER_DONE_INT_EN: 0 = DISABLE 1 = ENABLE
12	0x0	RXFIFO_DATA_REQ_INT_EN: 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
11	0x0	TXFIFO_DATA_REQ_INT_EN: 0 = DISABLE 1 = ENABLE
10	0x0	DGLITCH_INT_EN: 0 = DISABLE 1 = ENABLE
9	0x0	RXF_UNR_INT_EN: 0 = DISABLE 1 = ENABLE
8	0x0	TXF_OVF_INT_EN: 0 = DISABLE 1 = ENABLE
7	0x0	MEM_CMD_DONE_INT_EN: 0 = DISABLE 1 = ENABLE
6	0x0	ROM_CMD_DONE_INT_EN: 0 = DISABLE 1 = ENABLE
5	0x0	PRESENCE_DONE_INT_EN: 0 = DISABLE 1 = ENABLE
4	0x0	RESET_DONE_INT_EN: 0 = DISABLE 1 = ENABLE
3	0x0	ERR_CMD_INT_EN: 0 = DISABLE 1 = ENABLE
2	0x0	MEM_WR_ERR_INT_EN: 0 = DISABLE 1 = ENABLE
1	0x0	CRC_ERR_INT_EN: 0 = DISABLE 1 = ENABLE
0	0x0	PRESENCE_ERR_INT_EN: 0 = DISABLE 1 = ENABLE

24.3.11 OWR_INTR_STATUS_0

OWR Status Register

Offset: 028h | Read/Write: R/W | Reset: 0b0xx00000000000

Bit	R/W	Reset	Description
13	RW	0x0	BIT_TRANSFER_DONE: This bit is set when transfer of each bit done this is set on in one bit transfer mode software writes 1 to clear this bit 0 = NOT_DONE 1 = DONE
12	RO	X	RXFIFO_DATA_REQ: RX FIFO data req 0 = RX_NOT_RDY 1 = RX_RDY
11	RO	X	TXFIFO_DATA_REQ: TX FIFO data req 0 = TX_NOT_RDY 1 = TX_RDY

Bit	R/W	Reset	Description
10	RW	0x0	DGLITCH: This bit is set when data is not stable for at least 1us, Software writes a 1 to clear this bit. if deglitch detected data transfer should start from 1st. 0 = DGLITCH_NOT_DETECTED 1 = DGLITCH_DETECTED
9	RW	0x0	RXF_UNR: RX FIFO Under run: RO. This bit is set to 1 whenever software tries to read from an empty RX FIFO. Software writes a 1 to clear this bit. 0 = NOT_EMPTY 1 = EMPTY
8	RW	0x0	TXF_OVF: TX FIFO Overflow: RO. This bit is set to 1 whenever software tries to write to a full TX FIFO. Software writes a 1 to clear this bit. 0 = NOT_EMPTY 1 = EMPTY
7	RW	0x0	MEM_CMD_DONE: This Indicates the master has written data into eprom or data received from eprom without any error Software writes a 1 to clear it. 0 = NOT_DONE 1 = DONE
6	RW	0x0	ROM_CMD_DONE: This indicates master has received the rom data from battery Software writes a 1 to clear it. 0 = NOT_DONE 1 = DONE
5	RW	0x0	PRESENCE_DONE: This indicates the presence done, master has detected the device Software writes a 1 to clear it. 0 = NOT_DONE 1 = DONE
4	RW	0x0	RESET_DONE: This indicates the master has send the reset, then waits for presence Software writes a 1 to clear it. 0 = NOT_DONE 1 = DONE
3	RW	0x0	ERR_CMD: ERROR CMD: Indicates error command written in the register Software writes a 1 to clear it. 0 = CORRECT_CMD 1 = ERROR_CMD
2	RW	0x0	MEM_WR_ERR: MEM WR ERROR: Indicates the received data from eprom is correct or not Software writes a 1 to clear it. 0 = N0_ERROR 1 = ERROR
1	RW	0x0	CRC_ERR: CRC ERROR: Indicates the received data is correct or not Software writes a 1 to clear it. 0 = N0_ERROR 1 = ERROR
0	RW	0x0	PRESENCE_ERR: Presence ERROR. This bit is set when device presence not found 0 = SLAVE_DETECTED 1 = NO_SLAVE_DETECTED

24.3.12 OWR_INTR_SOURCE_0

This is a read-only register which returns the AND of Interrupt Source and Interrupt Mask registers.

OWR INTR Source Register

Offset: 02ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxx

Bit	Reset	Description
13	X	BIT_TRANSFER_DONE: 0 = NOT_DONE 1 = DONE

Bit	Reset	Description
12	X	RXFIFO_DATA_REQ: 0 = RX_NOT_RDY 1 = RX_RDY
11	X	TXFIFO_DATA_REQ: 0 = TX_NOT_RDY 1 = TX_RDY
10	X	DGLITCH: 0 = DGLITCH_NOT_DETECTED 1 = DGLITCH_DETECTED
9	X	RXF_UNR: 0 = NOT_EMPTY 1 = EMPTY
8	X	TXF_OVF: 0 = NOT_EMPTY 1 = EMPTY
7	X	MEM_CMD_DONE: 0 = NOT_DONE 1 = DONE
6	X	ROM_CMD_DONE: 0 = NOT_DONE 1 = DONE
5	X	PRESENCE_DONE: 0 = NOT_DONE 1 = DONE
4	X	RESET_DONE: 0 = NOT_DONE 1 = DONE
3	X	ERR_CMD: 0 = CORRECT_CMD 1 = ERROR_CMD
2	X	MEM_WR_ERR: 0 = N0_ERROR 1 = ERROR
1	X	CRC_ERR: 0 = N0_ERROR 1 = ERROR
0	X	PRESENCE_ERR: 0 = SLAVE_DETECTED 1 = NO_SLAVE_DETECTED

24.3.13 OWR_INTR_SET_0

This is a write-only register which can be used to set the interrupt status bit. A write to this register causes the bits in status register to be set 1 Read always returns 0.

Interrupt Set Register

Offset: 030h | Read/Write: R/W | Reset: 0b0xx00000000000

Bit	Reset	Description
13	0x0	BIT_TRANSFER_DONE: 0 = NOT_DONE 1 = DONE
10	0x0	DGLITCH: 0 = DGLITCH_NOT_DETECTED

Bit	Reset	Description
		1 = DGLITCH_DETECTED
9	0x0	RXF_UNR: 0 = NOT_EMPTY 1 = EMPTY
8	0x0	TXF_OVF: 0 = NOT_EMPTY 1 = EMPTY
7	0x0	MEM_CMD_DONE: 0 = NOT_DONE 1 = DONE
6	0x0	ROM_CMD_DONE: 0 = NOT_DONE 1 = DONE
5	0x0	PRESENCE_DONE: 0 = NOT_DONE 1 = DONE
4	0x0	RESET_DONE: 0 = NOT_DONE 1 = DONE
3	0x0	ERR_CMD: 0 = CORRECT_CMD 1 = ERROR_CMD
2	0x0	MEM_WR_ERR: 0 = N0_ERROR 1 = ERROR
1	0x0	CRC_ERR: 0 = N0_ERROR 1 = ERROR
0	0x0	PRESENCE_ERR: 0 = SLAVE_DETECTED 1 = NO_SLAVE_DETECTED

24.3.14 OWR_STATUS_0

OWR Status Register

Offset: 034h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxx00xxxxx

Bit	R/W	Reset	Description
23	RO	X	READ_SAMPLED_BIT: the bit is valid only RD_BUSY is cleared 0 = READ_ZERO 1 = READ_ONE
22	RO	X	RD_BUSY: READ : This is a self clearing bit and is cleared when read time slot completes on read sequence the sampled read bit is stored in READ_BIT 0 = IDLE 1 = BUSY
21	RO	X	WR1_BUSY: WRITE1 : This is a self clearing bit and is cleared when write one time slot completes on write sequence 1 is transferred 0 = IDLE 1 = BUSY
20	RO	X	WR0_BUSY: WRITE 0 : This bit is self clearing, and is cleared when write zero time slot completes on write sequence 0 is transferred 0 = IDLE 1 = BUSY
19	RO	X	RPP: this is sent reset is set(go), auto cleared on completion of reset

Bit	R/W	Reset	Description
			initialization sequence. 0 = IDLE 1 = RESET_PRESENCE_PULSE
18:13	RO	X	TX_FIFO_EMPTY_CNT: The number of slots that can be written to the tx fifo
12:7	RO	X	RX_FIFO_FULL_CNT: The number of slots to be read from the rx fifo
6	RW	0x0	RX_FLUSH: flush the rx fifo, cleared after fifo is empty 0 = DISABLE 1 = ENABLE
5	RW	0x0	TX_FLUSH: flush the tx fifo, cleared after fifo is empty 0 = DISABLE 1 = ENABLE
4	RO	X	RXF_EMPTY: RX FIFO empty status: RO. Hardware sets this bit to 1 if RX FIFO is empty Otherwise, this bit is set to 0. 0 = NOT_EMPTY 1 = EMPTY
3	RO	X	RXF_FULL: RX FIFO full status: RO. Hardware sets this bit to 1 if RX FIFO is full. Otherwise, this bit is set to 0. 0 = NOT_FULL 1 = FULL
2	RO	X	TXF_EMPTY: TX FIFO empty status: RO. Hardware sets this bit to 1 if TX FIFO is empty Otherwise, this bit is set to 0. 0 = NOT_EMPTY 1 = EMPTY
1	RO	X	TXF_FULL: TX FIFO full status: RO.Hardware sets this bit to 1 if TX FIFO is full. Otherwise, this bit is set to 0. 0 = NOT_FULL 1 = FULL
0	RO	X	RDY: Ready bit. This bit is set at the end of every transfer and its cleared by hardware when next transfer starts 0 = NOT_READY 1 = READY

24.3.15 OWR_CRC_0

OWR CRC Register

Offset: 038h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	CRC_CALC: CRC calculated by owr current wr/rd operation
15:0	X	CRC_RECEV: CRC Received on Read Data

24.3.16 OWR_BYTE_CNT_0

OWR BYTE_CNT Register

Offset: 03ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	TRANSMITTED: Number of bytes Transmitted on wr cmds or addr sent
15:0	X	RECEIVED: Number of bytes Received on Read Data includes crc byte cnt

24.3.17 OWR_TX_FIFO_0

OWR TX FIFO Register

Offset: 040h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	WR_DATA: TX FIFO

24.3.18 OWR_RX_FIFO_0

OWR RX FIFO Register

Offset: 044h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	RD_DATA: RX FIFO

25.0 SD/MMC CONTROLLER

The SDMMC Controller is capable of interfacing to SDMEM, SDIO, MMC, CE-ATA cards and to moviNAND and eMMC devices. The controller is hooked up to AHB bus and acts as both master and slave simultaneously. As a master on the bus it is capable of initiating data transfers between memory and external card, and as a slave it provides access to the configuration registers.

The Tegra® 2 Processor contains four identical instances of this controller.

Standards

- "CE-ATA Digital Protocol", CE-ATA Workgroup, Revision 1.0, 2-March-2005.
- "SD Specifications Part E1 SDIO Specification", Technical Committee SD Card Association, Version 2.00, 30-January 2007.
- "SD Specifications Part 1 Physical Layer Specification", Technical Committee SD Card Association, Version 2.00, 31-October 2005.
- "SD Specifications Part A2 SD Host Controller Standard Specification", Technical Committee SD Association, Version 2.00, 30-January 2007.
- JEDEC MMC/eMMC Electrical Interface Specification, Version 4.3", JEDEC SOLID STATE TECHNOLOGY ASSOCIATION, April 2007.

Features

- Adheres to SD Host Controller Standard Specification Version 2.0
- Supports MMC Specification Version 4.3
- Supports CE-ATA Digital Protocol revision 1.1
- Supports SD Memory Card Specification Version 2.0
- Supports SDIO Card Specification Version 2.0
- Supports eSD Specification Version 2.1
- Supports MMC Plus, MMC Mobile and Dual-Voltage MMC Cards.
- Support of 8-bit data interface for MMC cards
- Support SPI mode
- Allows card to interrupt host in 1 bit, 4-bit, 8-bit SD modes and SPI mode.
- Up to 200Mbits per second data rate using 4 parallel data lines (SD 4-bit mode) at 50MHz
- Up to 416Mbits per second data rate using 8 bit parallel data lines (MMC 8-bit mode) at 52MHz.
- Supports Read wait Control, Suspend/Resume operation for SDIO cards
- Supports FIFO overrun and underrun condition by stopping SD clock
- Supports Boot mode feature of MMC Specification 4.3 version.

25.1 Operation

SDMMC controller provides a register interface to software. These registers provide both control and observability. The host controller is dependent on the software for its initialization. The software develops card drivers to communicate and handle all types of memory cards.

25.1.1 Hardware / Software Partitioning

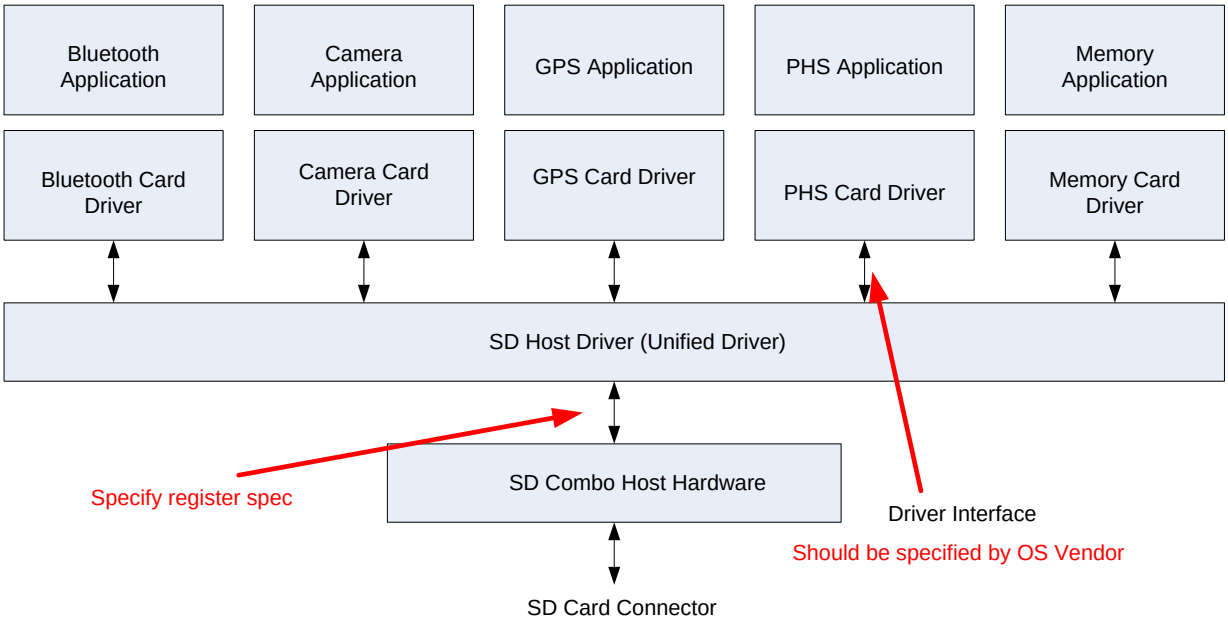
Hardware

Sends out the programmed command, stores the response and makes it visible to software. It updates the status registers and generates interrupts when attention is required. It also sends and receives data for a specified configuration programmed by software. The software only configures the memory address, amount of data and transfer direction; it is the hardware that takes care of fetching/storing data from/to memory.

Software

Determines which type of card is inserted in the slot by sending the initialization commands and observing the responses that are 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. Since the protocol constitutes of commands to the card and responses from the card, the software will have to program the controller with appropriate commands to communicate with the card.

Figure 66 Host Hardware and Driver Architecture



25.1.2 Performance

Table 81 Performance Maximums

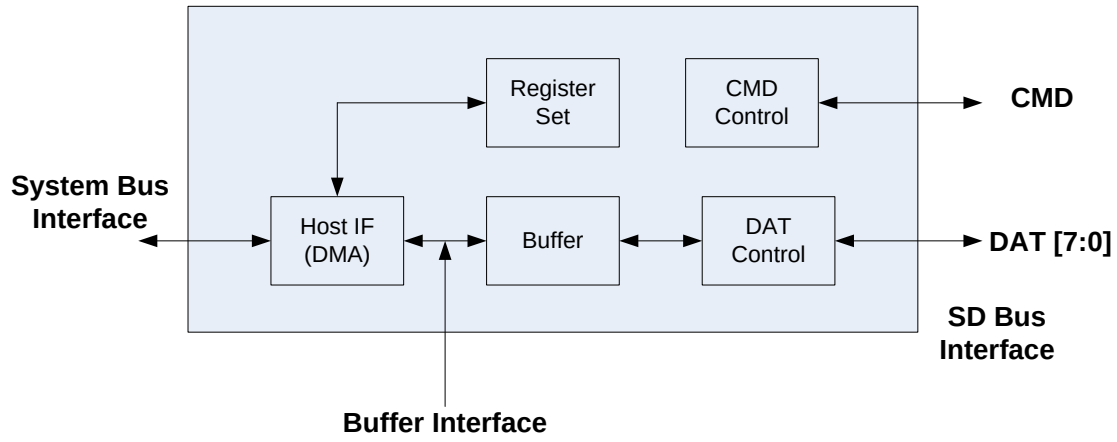
SPI	0-52MHz, Maximum data rate up to 52Mbits/sec
CE-ATA	0-52MHz, Maximum data rate up to 416Mbits/sec. Data bus width modes: 1bit (default), 4bit and 8bit.
MMC	0-52MHz, Maximum data rate up to 416Mbits/sec. bus width modes: 1bit (default), 4bit and 8bit.
SD Memory	0-50MHz, up to 200 Mbits/sec interface speed (using 4 parallel data lines).
SDIO	0-25MHZ i.e. 100 Mbits/sec interface speed (using 4 parallel data lines)

Caveats and Assumptions

- A single SDMMC controller handles one card at a time.
- The software should configure “bit 0” of the SDMMC_VENDOR_CLOCK_CNTRL_0 register before turning off the sdmmc_clk. This is required in order to support asynchronous card interrupts when there is no clock supplied to the card.
- In non-DMA mode of operation, the maximum block length supported is 512 bytes. Refer to the Programming Guidelines section for additional information.
- The PMC module monitors the card insertion/removal pin and data[1] line for interrupts when the chip is in LP0 state or when SDMMC power island is switched off.
- For Tegra 2, the software should check for card insertion and ensure whether it is write-protected or not, before configuring the controller. The card insertion/removal status and interrupts are void

25.2 Block Diagram

Figure 67. Block Diagram of Host Controller



The Host Controller has two bus interfaces, the System Bus Interface and the SD Bus Interface. The Host Controller assumes that these interfaces are asynchronous. The Host Driver is hooked to the system bus. The SD card is on SD Bus and operates on the clock going to the card. The Host Controller synchronizes signals to communicate between these interfaces. Blocks of data are synchronized at the buffer module.

25.3 Programming Guidelines

The guidelines described below adhere to “SD Host Controller Standard Specification Version 2.0”

25.3.1 High Speed Mode

1. Switch the card to High-Speed Mode.
2. Do not set the High Speed Bit in the SDMMC controller registers to 1.Only the card clock frequency should be increased to the required frequency.
3. While exiting High speed mode, decrease the card clock frequency and then switch the card to Normal mode.

25.3.2 Boot Mode

25.3.2.1 Boot Option1

1. Configure the clock going to the card to 20MHz.
2. Program the number of blocks. Configuring block length has no effect as it is always fixed to 512 bytes. Configuring data transfer direction has no impact, since boot mode is a CARD to HOST data transfer and is implicitly taken.
3. Configure to select ADMA1, ADMA2, and SDMA or PIO mode.
4. Configure SDMMC_VENDOR_BOOT_ACK_TIMEOUT_0 – The max timeout value to be programmed is 50ms.The programmed value does not include the 74cycles required to enter boot mode.
5. Configure SDMMC_VENDOR_BOOT_DAT_TIMEOUT_0 – The max timeout value to be programmed is 1 sec. The programmed value does not include the 74cycles required to enter boot mode.
6. Configure SDMMC_VENDOR_BOOT_CNTRL_0[1] to 1 to ask the engine to look for boot_ack.
7. Configure SDMMC_VENDOR_BOOT_CNTRL_0[0] to trigger the engine.
8. If SDMMC_INTERRUPT_STATUS_0[31] = 1, BOOT_ACK is not equal to 00101. The engine informs this to the software and continues to fetch data from the card.
9. If SDMMC_INTERRUPT_STATUS_0[30] = 1, it means BOOT_ACK timeout has occurred. The engine informs this to the software and continues to fetch data from the card.
10. The software looks for transfer complete interrupt. If data_time_out error occurs, the data transfer is not successful.
11. After successful data transfer from card, the software looks for sdma_engine_busy bit of SDMMC_OBS_BUS[4].Software does not program commands until this bit is zero.

25.3.2.2 Boot Option2

Boot option2 is treated like any normal read command. The BOOT_ACK should be enabled if the card supports boot acknowledgment. Program SDMMC_VENDOR_BOOT_ACK_TIMEOUT_0, and SDMMC_VENDOR_BOOT_DAT_TIMEOUT_0 based on the frequency of operation.

25.3.3 SDIO Data Transfer Abort

Asynchronous aborts can be applied at any point of transfer while reading data from the card, i.e. during block transfer or at block gap. But whereas data writes to card, aborts can only be issued at the block gap intervals. This is not a controller restriction, but a card restriction. To abort write transactions, software should keep Block Gap Request and then issue the Abort command.

25.3.4 Non-DMA Data Transfers

The depth of the FIFO is 128 words. The FIFO can only accommodate 128 entries (if a half-word is written into a slot instead of a word, it occupies one of the 128 slots available) and a maximum of 510 bytes can be stored in FIFO. It is assumed that only words are written into the FIFO.

If the block length in non-DMA is less than 128 words, programming sequence mentioned in the “SD Host Controller Specification” can be followed.

If the block length is greater than 512 bytes, software polls the buffer read/write enables in the present state register, and determines whether to further write to FIFO while writing to card, or read from FIFO while reading from card. The Buffer read/write ready interrupts are only given on a block by block basis.

Data transfer with the card stalls if the programmed bytes are not written in FIFO while writing to card or if the FIFO is full while reading from the card.

25.4 SDMMC Registers

25.4.1 SDMMC_SYSTEM_ADDRESS_0

ADDRESS

This register contains the system memory address for a SDMA transfer. When the Host Controller stops a SDMA transfer, this register shall point to the system address of the next contiguous data position. It can be accessed only if no transaction is executing (i.e., after a transaction has stopped). Read operations during transfers may return an invalid value.

The Host Driver shall initialize this register before starting a SDMA transaction. After SDMA has stopped, the next system address of the next contiguous data position can be read from this register. The SDMA transfer waits at the every boundary specified by the Host SDMA Buffer Boundary in the Block Size register. The Host Controller generates DMA Interrupt to request the Host Driver to update this register. The Host Driver sets the next system address of the next data position to this register. When the most upper byte of this register (003h) is written, the Host Controller restarts the SDMA transfer.

When restarting SDMA by the Resume command or by setting Continue Request in the Block Gap Control register, the Host Controller shall start at the next contiguous address stored here in the SDMA System Address register. ADMA does not use this register.

SDMA System Address Register

Offset: 000h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	ADDRESS

25.4.2 SDMMC_BLOCK_SIZE_BLOCK_COUNT_0

HOST_DMA_BUFFER_SIZE

The large contiguous memory space may not be available in the virtual memory system. To perform long SDMA transfer, SDMA System Address register shall be updated at every system memory boundary during SDMA transfer.

These bits specify the size of contiguous buffer in the system memory. The SDMA transfer shall wait at the every boundary specified by these fields and the Host Controller generates the DMA Interrupt to request the Host Driver to update the SDMA System Address register. At the end of transfer, the Host Controller may issue or may not issue DMA Interrupt. In particular, DMA Interrupt shall not be issued after Transfer Complete Interrupt is issued. In case of this register is set to 0 (buffer size = 4K bytes), lower 12-bit of byte address points data in the contiguous buffer and the upper 20-bit points the location of the buffer in the system memory. The SDMA transfer stops when the Host Controller detects carry out of the address from bit 11 to 12.

These bits shall be supported when the SDMA Support in the Capabilities register is set to 1 and this function is active when the DMA Enable in the Transfer Mode register is set to 1.

ADMA does not use this register.

000b 4K bytes (Detects A11 carry out)

001b 8K bytes (Detects A12 carry out)

010b 16K Bytes (Detects A13 carry out)

011b 32K Bytes (Detects A14 carry out)

100b 64K bytes (Detects A15 carry out)

101b 128K Bytes (Detects A16 carry out)

110b 256K Bytes (Detects A17 carry out)

111b 512K Bytes (Detects A18 carry out)

XFER_BLOCK_SIZE_11_0

This register specifies the block size of data transfers for CMD17, CMD18, CMD24, CMD25, and CMD53. Values ranging from 1 up to the maximum buffer size can be set. In case of memory, it shall be set up to 512 bytes (Refer to Implementation Note in Section 1.7.2). It can be accessed only if no transaction is executing (i.e., after a transaction has stopped). Read operations during transfers may return an invalid value, and write operations shall be ignored.

0800h 2048 Bytes

0200h 512 Bytes

01FFh 511 Bytes

0004h 4 Bytes

0003h 3 Bytes

0002h 2 Bytes

0001h 1 Byte

0000h No data transfer

BLOCKS_COUNT

This register is enabled when Block Count Enable in the Transfer Mode register is set to 1 and is valid only for multiple block transfers. The Host Driver shall set this register to a value between 1 and the maximum block count. The Host Controller decrements the block count after each block transfer and stops when the count reaches zero. Setting the block count to 0 results in no data blocks is transferred.

This register should be accessed only when no transaction is executing (i.e., after transactions are stopped). During data transfer, read operations on this register may return an invalid value and write operations are ignored. When a suspend command is completed, the number of blocks yet to be transferred can be determined by reading this register. Before issuing a resume command, the Host Driver shall restore the previously saved block count.

FFFFh 65535 blocks

0002h 2 blocks

0001h 1 block

0000h Stop Count

XFER_BLOCK_SIZE_12

Transfer Block Size 12th bit. This bit is added to support 4Kb Data block transfer.

Block Size Register

Offset: 004h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	BLOCKS_COUNT
15	0x0	XFER_BLOCK_SIZE_12
14:12	0x0	HOST_DMA_BUFFER_SIZE: 0 = DMA4K 1 = DMA8K 2 = DMA16K 3 = DMA32K 4 = DMA64K 5 = DMA128K 6 = DMA256K

Bit	Reset	Description
		7 = DMA512K
11:0	0x0	XFER_BLOCK_SIZE_11_0

25.4.3 SDMMC_ARGUMENT_0

COMMAND_ARGUMENT

The SD Command Argument is specified as bit39-8 of Command-Format in the Physical Layer Specification.

Command Argument Register

Offset: 008h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	COMMAND_ARGUMENT

25.4.4 SDMMC_CMD_XFER_MODE_0

COMMAND_INDEX

These bits shall be set to the command number (CMD0-63, ACMD0-63) that is specified in bits 45-40 of the Command-Format in the Physical Layer Specification and SDIO Card Specification.

COMMAND_TYPE

There are three types of special commands: Suspend, Resume and Abort. These bits shall be set to 00b for all other commands.

- Suspend Command

  If the Suspend command succeeds, the Host Controller shall assume the SD Bus has been released and that it is possible to issue the next command, which uses the DAT line. The Host Controller shall de-assert Read Wait for read transactions and stop checking busy for write transactions. The interrupt cycle shall start, in 4-bit mode. If the Suspend command fails, the Host Controller shall maintain its current state, and the Host Driver shall restart the transfer by setting Continue Request in the Block Gap Control register.
- Resume Command

  The Host Driver re-starts the data transfer by restoring the registers in the range of 000-00Dh. The Host Controller shall check for busy before starting write transfers.
- Abort Command

  If this command is set when executing a read transfer, the Host Controller shall stop reads to the buffer. If this command is set when executing a write transfer, the Host Controller shall stop driving the DAT line. After issuing the Abort command, the Host Driver should issue a software reset.

  11b Abort CMD12, CMD52 for writing "I/O Abort" in CCCR

  10b Resume CMD52 for writing "Function Select" in CCCR

  01b Suspend CMD52 for writing "Bus Suspend" in CCCR

  00b Normal Other commands

DATA_PRESENT_SELECT

This bit is set to 1 to indicate that data is present and shall be transferred using the DAT line. It is set to 0 for the following:

- Commands using only CMD line (ex. CMD52).
- Commands with no data transfer but using busy signal on DAT[0] line (R1b or R5b ex. CMD38)

- Resume command

CMD_INDEX_CHECK_EN

If this bit is set to 1, the Host Controller shall check the Index field in the response to see if it has the same value as the command index. If it is not, it is reported as a Command Index Error. If this bit is set to 0, the Index field is not checked.

CMD_CRC_CHECK_EN

If this bit is set to 1, the Host Controller shall check the CRC field in the response. If an error is detected, it is reported as a Command CRC Error. If this bit is set to 0, the CRC field is not checked. The position of CRC field is determined according to the length of the response.

RESP_TYPE_SELECT

Normal Mode:

- 00 No Response
- 01 Response Length 136
- 10 Response Length 48
- 11 Response Length 48 check Busy after response

SPI Mode:(VENDOR Definition)

- 00 R1 Response Length 8
- 01 R2 Response Length 16
- 10 R3 Response Length 40
- 11 R1b Response Length 8

SPI_MODE(VENDOR Bit)

SPI Mode is enabled. The Response types definitions are different. Refer to Response type field.

CMD_COMP_ATA- CEATA Command Completion Signal Enable.(VENDOR Bit)

Enabled to wait for CCS Signal. After this has been received the Transfer complete occurs.

MULTI_BLOCK_SELECT - Multi / Single Block Select

This bit is set when issuing multiple-block transfer commands using DAT line. For any other commands, this bit shall be set to 0. If this bit is 0, it is not necessary to set the Block Count register.

1 Multiple Block

0 Single Block

DATA_XFER_DIR_SEL - Data Transfer Direction Select

This bit defines the direction of DAT line data transfers. The bit is set to 1 by the Host Driver to transfer data from the SD card to the SD Host Controller and it is set to 0 for all other commands.

1 Read (Card to Host)

0 Write (Host to Card)

AUTO_CMD12_EN - Auto CMD12 Enable

Multiple block transfers for memory require CMD12 to stop the transaction. When this bit is set to 1, the Host Controller shall issue CMD12 automatically when last block transfer is completed. The Host Driver shall not set this bit to issue commands that do not require CMD12 to stop data transfer. In particular, secure commands defined in the Part 3 File Security specification do not require CMD12.

BLOCK_COUNT_EN - Block Count Enable

This bit is used to enable the Block Count register, which is only relevant for multiple block transfers. When this bit is 0, the Block Count register is disabled, which is useful in executing an infinite transfer. (Refer to Table 2-8) If ADMA2 data transfer is more than 65535 blocks, this bit shall be set to 0. In this case, data transfer length is designated by Descriptor Table.

DMA_EN - DMA Enable

This bit enables DMA functionality as described in section 1.4. DMA can be enabled only if it is supported as indicated in the Capabilities register. One of the DMA modes can be selected by DMA Select in the Host Control register. If DMA is not supported, this bit is meaningless and shall always read 0. If this bit is set to 1, a DMA operation shall begin when the Host Driver writes to the upper byte of Command register (00Fh).

Command and Transfer Mode Register

Offset: 00ch | Read/Write: R/W | Reset: 0b00000000000x00xxxxxxxx0000x000

Bit	Reset	Description
29:24	0x0	COMMAND_INDEX
23:22	0x0	COMMAND_TYPE: 0 = NORMAL 1 = SUSPEND 2 = RESUME 3 = ABORT
21	0x0	DATA_PRESENT_SELECT: 0 = NO_DATA_TRANSFER 1 = DATA_TRANSFER
20	0x0	CMD_INDEX_CHECK_EN: 0 = DISABLE 1 = ENABLE
19	0x0	CMD_CRC_CHECK_EN: 0 = DISABLE 1 = ENABLE
17:16	0x0	RESP_TYPE_SELECT: 0 = NO_RESPONSE 1 = RESP_LENGTH_136 2 = RESP_LENGTH_48 3 = RESP_LENGTH_48BUSY 0 = SPI_R1_RESPONSE 1 = SPI_R2_RESPONSE 2 = SPI_R3_RESPONSE 3 = SPI_R1b_RESPONSE
7	0x0	SPI_MODE: 0 = DISABLE 1 = ENABLE
6	0x0	CMD_COMP_ATA: 0 = DISABLE 1 = ENABLE
5	0x0	MULTI_BLOCK_SELECT: 0 = DISABLE 1 = ENABLE
4	0x0	DATA_XFER_DIR_SEL: 0 = WRITE 1 = READ
2	0x0	AUTO_CMD12_EN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1	0x0	BLOCK_COUNT_EN: 0 = DISABLE 1 = ENABLE
0	0x0	DMA_EN: 0 = DISABLE 1 = ENABLE

25.4.5 SDMMC_RESPONSE_R0_R1_0

Command Response Registers

The Table below describes the mapping of command responses from the SD Bus to this register for each response type. In the table, R[…] refers to a bit range within the response data as transmitted on the SD Bus, REP[] refers to a bit range within the Response register.

Kind of Response	Meaning of Response	Response Field	Response Register
R1, R1b (normal response)	Card Status	R [39:8]	REP [31:0]
R1b (Auto CMD12 response)	Card Status for Auto CMD12	R [39:8]	REP [127:96]
R2 (CID, CSD register)	CID or CSD reg. incl.	R [127:8]	REP [119:0]
R3 (OCR register)	OCR register for memory	R [39:8]	REP [31:0]
R4 (OCR register)	OCR register for I/O etc	R [39:8]	REP [31:0]
R5,R5b	SDIO response	R [39:8]	REP [31:0]
R6 (Published RCA response)	New published RCA[31:16] etc	R [39:8]	REP [31:0]

Response Bit Definition for Each Response Type

Command Response [31:0] (R0) Register

Offset: 010h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	CMD_RESP_31_16
15:0	X	CMD_RESP_15_0

25.4.6 SDMMC_RESPONSE_R2_R3_0

Command Response [63:32] (R2) Register

Offset: 014h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	CMD_RESP_63_48
15:0	X	CMD_RESP_47_32

25.4.7 SDMMC_RESPONSE_R4_R5_0

Command Response [95:64] (R4) Register

Offset: 018h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	CMD_RESP_95_80

Bit	Reset	Description
15:0	X	CMD_RESP_79_64

25.4.8 SDMMC_RESPONSE_R6_R7_0

Command Response [127:96] (R6) Register

Offset: 01ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	CMD_RESP_127_112
15:0	X	CMD_RESP_111_96

25.4.9 SDMMC_BUFFER_DATA_PORT_0

The Host Controller Buffer can be accessed through this 32-bit Data Port Register.

Buffer Data Port Register

Offset: 020h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	BUFFER_DATA:

25.4.10 SDMMC_PRESENT_STATE_0

DAT_7_4_LINE_LEVEL - DAT[7:4] Line Signal Level(VENDOR Bit)

This status is used to check the DAT line level to recover from errors, and for debugging.

CMD_LINE_LEVEL - CMD Line Signal Level

This status is used to check the CMD line level to recover from errors, and for debugging.

DAT_3_0_LINE_LEVEL - DAT[3:0] Line Signal Level

This status is used to check the DAT line level to recover from errors, and for debugging. This is especially useful in detecting the busy signal level from DAT[0].

WRITE_PROTECT_LEVEL - Write Protect Switch Pin Level

CARD_DETECT_PIN_LEVEL - Card Detect Pin Level

CARD_STATE_STABLE - Card State Stable

CARD_INSERTED - Card Inserted

BUFFER_READ_EN - Buffer Read Enable

This status is used for non-DMA read transfers.

BUFFER_WRITE_EN - Buffer Write Enable

This status is used for non-DMA write transfers.

READ_XFER_ACTIVE - Read Transfer Active

WRITE_XFER_ACTIVE - Write Transfer Active

DAT_LINE_ACTIVE - DAT Line Active

CMD_INHIBIT_DAT - Command Inhibit (DAT)

CMD_INHIBIT_CMD - Command Inhibit (CMD)

Present State Register

Offset: 024h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:25	X	DAT_7_4_LINE_LEVEL
24	X	CMD_LINE_LEVEL: 0 = LOW 1 = HIGH
23:20	X	DAT_3_0_LINE_LEVEL
19	X	WRITE_PROTECT_LEVEL: 0 = PROTECTED 1 = ENABLED
18	X	CARD_DETECT_PIN_LEVEL: 0 = NO_CARD 1 = CARD
17	X	CARD_STATE_STABLE: 0 = DEBOUNCE 1 = INSERTED
16	X	CARD_INSERTED: 0 = DEBOUNCE 1 = INSERTED
11	X	BUFFER_READ_EN: 0 = DISABLE 1 = ENABLE
10	X	BUFFER_WRITE_EN: 0 = DISABLE 1 = ENABLE
9	X	READ_XFER_ACTIVE: 0 = NO_DATA 1 = TRANSFERING
8	X	WRITE_XFER_ACTIVE: 0 = NO_DATA 1 = TRANSFERING
2	X	DAT_LINE_ACTIVE: 0 = INACTIVE 1 = ACTIVE
1	X	CMD_INHIBIT_DAT: 0 = INACTIVE 1 = ACTIVE
0	X	CMD_INHIBIT_CMD: 0 = INACTIVE 1 = ACTIVE

25.4.11 SDMMC_POWER_CONTROL_HOST_0

WAKEUP_ON_CARD_REMOVAL - Wakeup Event Enable On SD Card Removal

This bit enables wakeup event via Card Removal assertion in the Normal Interrupt Status register. FN_WUS (Wake Up Support) in CIS does not affect this bit.

WAKEUP_ON_CARD_INSERTION - Wakeup Event Enable On SD Card Insertion

This bit enables wakeup event via Card Insertion assertion in the Normal Interrupt Status register. FN_WUS (Wake Up Support) in CIS does not affect this bit.

WAKEUP_ON_CARD_INTERRUPT - Wakeup Event Enable On Card Interrupt

This bit enables wakeup event via Card Interrupt assertion in the Normal Interrupt Status register. This bit can be set to 1 if FN_WUS (Wake Up Support) in CIS is set to 1.

INTERRUPT_AT_BLOCK_GAP - Interrupt At Block Gap

This bit is valid only in 4-bit mode of the SDIO card and selects a sample point in the interrupt cycle. Setting to 1 enables interrupt detection at the block gap for a multiple block transfer. Setting to 0 disables interrupt detection during a multiple block transfer. If the SD card cannot signal an interrupt during a multiple block transfer, this bit should be set to 0. When the Host Driver detects an SD card insertion, it shall set this bit according to the CCCR of the SDIO card.

READ_WAIT_CONTROL - Read Wait Control

The read wait function is optional for SDIO cards. If the card supports read wait, set this bit to enable use of the read wait protocol to stop read data using the DAT[2] line. Otherwise, the Host Controller has to stop the SD Clock to hold read data, which restricts commands generation. When the Host Driver detects an SD card insertion, it shall set this bit according to the CCCR of the SDIO card. If the card does not support read wait, this bit shall never be set to 1 otherwise DAT line conflict may occur. If this bit is set to 0, Suspend/Resume cannot be supported.

CONTINUE_REQUEST - Continue Request

This bit is used to restart a transaction, which was stopped using the Stop At Block Gap Request. To cancel stop at the block gap, set Stop At Block Gap Request to 0 and set this bit 1 to restart the transfer. The Host Controller automatically clears this bit in either of the following cases:

- In the case of a read transaction, the DAT Line Active changes from 0 to 1 as a read transaction restarts.
- In the case of a write transaction, the Write Transfer Active changes from 0 to 1 as the write transaction restarts. Therefore, it is not necessary for Host Driver to set this bit to 0. If Stop At Block Gap Request is set to 1, any write to this bit is ignored.

STOP_AT_BLOCK_GAP_REQUEST - Stop At Block Gap Request

This bit is used to stop executing read and write transaction at the next block gap for non-DMA, SDMA and ADMA transfers. The Host Driver shall leave this bit set to 1 until the Transfer Complete is set to 1. Clearing both Stop At Block Gap Request and Continue Request shall not cause the transaction to restart. When Host Controller version is 1.00, the Host Driver can set this bit if the card supports Read Wait Control. When Host Controller version is 2.00 or higher, the Host Driver can set this bit regardless of the card supports Read Wait Control. The Host Controller shall stop read transfer by using Read Wait or stopping SD clock.

In case of write transfers in which the Host Driver writes data to the Buffer Data Port register, the Host Driver shall set this bit after all block data is written. If this bit is set to 1, the Host Driver shall not write data to Buffer Data Port register. This bit affects Read Transfer Active, Write Transfer Active, DAT Line Active and Command Inhibit (DAT) in the Present State register. Regarding detailed control of bits D01 and D00.

SD_BUS_VOLTAGE_SELECT - SD Bus Voltage Select

By setting these bits, the Host Driver selects the voltage level for the SD card. Before setting this register, the Host Driver shall check the Voltage Support bits in the Capabilities register. If an unsupported voltage is selected, the Host System shall not supply SD Bus voltage.

111b 3.3V (Typ.)

110b 3.0V (Typ.)

101b 1.8V (Typ.)

100b 000b Reserved

SD_BUS_POWER - SD Bus Power

Before setting this bit, the SD Host Driver shall set SD Bus Voltage Select. If the Host Controller detects the No Card state, this bit shall be cleared. If this bit is cleared, the Host Controller shall immediately stop driving CMD and DAT[3:0] (tri-state) and drive SDCLK to low level.

CARD_DETECT_SIGNAL_DETECT - Card Detect Signal Selection

This bit selects source for the card detection. When the source for the card detection is switched, the interrupt should be disabled during the switching period by clearing the Interrupt Status/Signal Enable register in order to mask unexpected interrupt being caused by the glitch. The Interrupt Status/Signal Enable should be disabled during over the period of debouncing.

CARD_DETECT_TEST_LVL - Card Detect Test Level

This bit is enabled while the Card Detect Signal Selection is set to 1 and it indicates card inserted or not.

EXTENDED_DATA_TRANSFER_WIDTH (VENDOR Bit)

1:8-bit Mode,DATA_XFER_WIDTH is ignored.

0:Card Bust is as per DATA_XFER_WIDTH value

DMA_SELECT - DMA Select

One of supported DMA modes can be selected. The host driver shall check support of DMA modes by referring the Capabilities register. Use of selected DMA is determined by DMA Enable of the Transfer Mode register.

00 SDMA is selected

01 32-bit Address ADMA1 is selected(VENDOR Definition)

10 32-bit Address ADMA2 is selected

11 64-bit Address ADMA2 is selected (VENDOR Definition: 1xh is considered as ADMA2.)

HIGH_SPEED_EN - High Speed Enable

This bit is optional. Before setting this bit, the Host Driver shall check the High Speed Support in the Capabilities register. If this bit is set to 0 (default), the Host Controller outputs CMD line and DAT lines at the falling edge of the SD Clock (up to 25MHz). If this bit is set to 1, the Host Controller outputs CMD line and DAT lines at the rising edge of the SD Clock (up to 50MHz).

DATA_XFER_WIDTH - Data Transfer Width

This bit selects the data width of the Host Controller. The Host Driver shall set it to match the data width of the SD card.

LED_CONTROL - LED Control

This bit is used to caution the user not to remove the card while the SD card is being accessed. If the software is going to issue multiple SD commands, this bit can be set during all these transactions. It is not necessary to change for each transaction.

Power Control / Host Control Register

Offset: 028h | Read/Write: R/W | Reset: 0b000xxxx0000xxxx000000000000

Bit	Reset	Description
26	0x0	WAKEUP_ON_CARD_REMOVAL: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
25	0x0	WAKEUP_ON_CARD_INSERTION: 0 = DISABLE 1 = ENABLE
24	0x0	WAKEUP_ON_CARD_INTERRUPT: 0 = DISABLE 1 = ENABLE
19	0x0	INTERRUPT_AT_BLOCK_GAP: 0 = DISABLE 1 = ENABLE
18	0x0	READ_WAIT_CONTROL: 0 = DISABLE 1 = ENABLE
17	0x0	CONTINUE_REQUEST: 0 = IGNORED 1 = RESTART
16	0x0	STOP_AT_BLOCK_GAP_REQUEST: 0 = STOP 1 = TRANSFER
11:9	0x0	SD_BUS_VOLTAGE_SELECT: 5 = V1_8 6 = V3_0 7 = V3_3
8	0x0	SD_BUS_POWER: 0 = POWER_OFF 1 = POWER_ON
7	0x0	CARD_DETECT_SIGNAL_DETECT: 0 = SDCD 1 = CARD_DTECT_TST_LVL
6	0x0	CARD_DETECT_TEST_LVL: 0 = NO_CARD 1 = CARD_INSERTED
5	0x0	EXTENDED_DATA_TRANSFER_WIDTH: 0 = NOBIT_8 1 = BIT_8
4:3	0x0	DMA_SELECT: 0 = SDMA 1 = ADMA1_32BIT 2 = ADMA2_32BIT 3 = ADMA2_64BIT
2	0x0	HIGH_SPEED_EN: 0 = NORMAL_SPEED 1 = HIGH_SPEED
1	0x0	DATA_XFER_WIDTH: 0 = BIT_1 1 = BIT_4
0	0x0	LED_CONTROL: 0 = OFF 1 = ON

25.4.12 SDMMC_SW_RESET_TIMEOUT_CTRL_CLOCK_CONTROL_0

SW_RESET_FOR_DAT_LINE - Software Reset For DAT Line

Only part of data circuit is reset. DMA circuit is also reset. The following registers and bits are cleared by this bit:

- Buffer Data Port register

- Buffer is cleared and initialized.

- Present State register
  - Buffer Read Enable
  - Buffer Write Enable
  - Read Transfer Active
  - Write Transfer Active
  - DAT Line Active
  - Command Inhibit (DAT)
- Block Gap Control register
  - Continue Request
  - Stop At Block Gap Request
- Normal Interrupt Status register
  - Buffer Read Ready
  - Buffer Write Ready
  - DMA Interrupt
  - Block Gap Event
  - Transfer Complete

SW_RESET_FOR_CMD_LINE - Software Reset For CMD Line

Only part of command circuit is reset. The following registers and bits are cleared by this bit:

- Present State register
  - Command Inhibit (CMD)
- Normal Interrupt Status register
  - Command Complete

SW_RESET_FOR_ALL - Software Reset For All

This reset affects the entire Host Controller except for the card detection circuit. Register bits of type ROC, RW, RW1C, RWAC are cleared to 0. During its initialization, the Host Driver shall set this bit to 1 to reset the Host Controller. The Host Controller shall reset this bit to 0 when Capabilities registers are valid and the Host Driver can read them. Additional use of Software Reset for All may not affect the value of the Capabilities registers. If this bit is set to 1, the host driver should issue reset command and reinitialize the SD card.

DATA_TIMEOUT_COUNTER_VALUE - Data Timeout Counter Value

This value determines the interval by which DAT line timeouts are detected. For more information about timeout generation, refer to the Data Timeout Error in the Error Interrupt Status register. Timeout clock frequency will be generated by dividing the base clock TMCLK value by this value. When setting this register, prevent inadvertent timeout events by clearing the Data Timeout Error Status Enable (in the Error Interrupt Status Enable register)

- 1111b Reserved
- 1110b TMCLK x pow(2,27)
- 0001b TMCLK x pow(2,14)
- 0000b TMCLK x pow(2,13)

SDCLK_FREQUENCYSELECT - SDCLK Frequency Select

This register is used to select the frequency of SDCLK pin. The frequency is not programmed directly; rather this register holds the divisor of the Base Clock Frequency For SD Clock in the Capabilities register. Only the following settings are allowed.

- SD_CLOCK_EN - SD Clock Enable

  The Host Controller shall stop SDCLK when writing this bit to 0. SDCLK Frequency Select can be changed when this bit is 0. Then, the Host Controller shall maintain the same clock frequency until SDCLK is stopped (Stop at SDCLK=0). If the Card Inserted in the Present State register is cleared, this bit shall be cleared.

- INTERNAL_CLOCK_STABLE - Internal Clock Stable

  This bit is set to 1 when SD Clock is stable after writing to Internal Clock Enable in this register to 1. The SD Host Driver shall wait to set SD Clock Enable until this bit is set to 1.

  Note: This is useful when using PLL for a clock oscillator that requires setup time.

- INTERNAL_CLOCK_EN - Internal Clock Enable

  This bit is set to 0 when the Host Driver is not using the Host Controller or the Host Controller awaits a wakeup interrupt. The Host Controller should stop its internal clock to go very low power state. Still, registers shall be able to be read and written. Clock starts to oscillate when this bit is set to 1. When clock oscillation is stable, the Host Controller shall set Internal Clock Stable in this register to 1. This bit shall not affect card detection.

Clock Control Register

Offset: 02ch | Read/Write: R/W | Reset: 0b000xxxx000000000000xxxxx0x0

Bit	R/W	Reset	Description
26	RW	0x0	SW_RESET_FOR_DAT_LINE: 0 = WORK 1 = RESETED
25	RW	0x0	SW_RESET_FOR_CMD_LINE: 0 = WORK 1 = RESETED
24	RW	0x0	SW_RESET_FOR_ALL: 0 = WORK 1 = RESETED
19:16	RW	0x0	DATA_TIMEOUT_COUNTER_VALUE
15:8	RW	0x0	SDCLK_FREQUENCYSELECT: 128 = DIV256 64 = DIV128 32 = DIV64 16 = DIV32 8 = DIV16 4 = DIV8 2 = DIV4 1 = DIV2 0 = BASE
2	RW	0x0	SD_CLOCK_EN: 0 = DISABLE 1 = ENABLE
1	RO	X	INTERNAL_CLOCK_STABLE: 0 = NOT_READY 1 = READY
0	RW	0x0	INTERNAL_CLOCK_EN: 0 = STOP 1 = OSCILLATE

25.4.13 SDMMC_INTERRUPT_STATUS_0

VEND_SPEC_ERR[1:0]

1:BOOT_ACK_ERR - Occurs When Boot Ack Status is not equal to '010'

0:BOOT_ACK_TIMEOUT_ERR - Occurs When Boot Ack is not received within the programmed number of cycles.

CEATA_ERROR

Occurs when the ATA command termination has occurred.

TARGET_RESP_ERROR - Not supported for Tegra 2 Processor

SPI_ERR

Indicate when SPI Error has occurred. The SPI Errors are registered in SPI_INTERRUPT_STATUS register.

ADMA_ERR

Occurs when detecting that one of the bits in Auto CMD12 Error Status register has changed from 0 to 1. This bit is set to 1,not only when the errors in Auto CMD12 occur but also when Auto CMD12 is not executed due to the previous command error.

AUTO_CMD12_ERR

Occurs when detecting that one of the bits in Auto CMD12 Error Status register has changed from 0 to 1. This bit is set to 1,not only when the errors in Auto CMD12 occur but also when Auto CMD12 is not executed due to the previous command error.

CURRENT_LIMIT_ERR

By setting the SD Bus Power bit in the Power Control register, the Host Controller is requested to supply power for the SD Bus. If the Host Controller supports the Current Limit function, it can be protected from an illegal card by stopping power supply to the card in which case this bit indicates a failure status. Reading 1 means the Host Controller is not supplying power to SD card due to some failure. Reading 0 means that the Host Controller is supplying power and no error has occurred.

The Host Controller may require some sampling time to detect the current limit. If the Host Controller does not support this function, this bit shall always be set to 0.

DATA_END_BIT_ERR

Occurs either when detecting 0 at the end bit position of read data which uses the DAT line or at the end bit position of the CRC Status.

DATA_CRC_ERR

Occurs when detecting CRC error when transferring read data which uses the DAT line or when detecting the Write CRC status having a value of other than "010".

DATA_TIMEOUT_ERR

Occurs when detecting one of following timeout conditions.

- Busy timeout for R1b,R5b type
- Busy timeout after Write CRC status
- Write CRC Status timeout
- Read Data timeout.

COMMAND_INDEX_ERR

Occurs if a Command Index error occurs in the command response.

COMMAND_END_BIT_ERR

Occurs when detecting that the end bit of a command response is 0.

COMMAND_CRC_ERR

Command CRC Error is generated in two cases.

- If a response is returned and the Command Timeout Error is set to 0 (indicating no timeout), this bit is set to 1 when detecting a CRC error in the command response.
- The Host Controller detects a CMD line conflict by monitoring the CMD line when a command is issued. If the Host Controller drives the CMD line to 1 level, but detects 0 level on the CMD line at the next SDCLK edge, then the Host Controller shall abort the command (Stop driving CMD line) and set this bit to 1. The Command Timeout Error shall also be set to 1 to distinguish CMD line conflict

COMMAND_TIMEOUT_ERR

Occurs only if no response is returned within 64 SDCLK cycles from the end bit of the command. If the Host Controller detects a CMD line conflict, in which case Command CRC Error shall also be set as shown in Table 2-25, this bit shall be set without waiting for 64 SDCLK cycles because the command will be aborted by the Host Controller.

ERR_INTERRUPT

If any of the bits in the Error Interrupt Status register are set, then this bit is set. Therefore the Host Driver can efficiently test for an error by checking this bit first. This bit is read only.

CARD_INTERRUPT

Writing this bit to 1 does not clear this bit. It is cleared by resetting the SD card interrupt factor. In 1-bit mode, the Host Controller shall detect the Card Interrupt without SD Clock to support wakeup. In 4-bit mode, the card interrupt signal is sampled during the interrupt cycle, so there are some sample delays between the interrupt signal from the SD card and the interrupt to the Host System. It is necessary to define how to handle this delay.

CARD_REMOVAL

This status is set if the Card Inserted in the Present State register changes from 1 to 0. When the Host Driver writes this bit to 1 to clear this status, the status of the Card Inserted in the Present State register should be confirmed. Because the card detect state may possibly be changed when the Host Driver clear this bit and interrupt event may not be generated.

CARD_INSERTION

This status is set if the Card Inserted in the Present State register changes from 0 to 1. When the Host Driver writes this bit to 1 to clear this status, the status of the Card Inserted in the Present State register should be confirmed. Because the card detect state may possibly be changed when the Host Driver clear this bit and interrupt event may not be generated.

BUFFER_READ_READY

This status is set if the Buffer Read Enable changes from 0 to 1. Refer to the Buffer Read Enable in the Present State register.

BUFFER_WRITE_READY

This status is set if the Buffer Write Enable changes from 0 to 1. Refer to the Buffer Write Enable in the Present State register.

DMA_INTERRUPT

This status is set if the Host Controller detects the Host DMA Buffer boundary during transfer. Refer to the Host DMA Buffer Boundary in the Block Size register. Other DMA interrupt factors may be added in the future. This interrupt shall not be generated after the Transfer Complete.

BLOCK_GAP_EVENT

If the Stop At Block Gap Request in the Block Gap Control register is set, this bit is set when both a read / write transaction is stopped at a block gap. If Stop At Block Gap Request is not set to 1, this bit is not set to 1.

XFER_COMPLETE

This bit is set when a read / write transfer is completed.

- In the case of a Read Transaction
- In the case of a Write Transaction

CMD_COMPLETE

This bit is set when get the end bit of the command response. (Except Auto CMD12) Refer to Command Inhibit (CMD) in the Present State register.

Normal Interrupt Status Register

Offset: 030h | Read/Write: R/W | Reset: 0b00000x0000000000xxxxxxxx00000000

Bit	R/W	Reset	Description
31:30	RW	0x0	VEND_SPEC_ERR: 0 = DISABLE 3 = ENABLE
29	RW	0x0	CEATA_ERROR: 0 = NO_ERROR 1 = ERROR
28	RW	0x0	TARGET_RESP_ERROR: 0 = NO_ERROR 1 = ERROR
27	RW	0x0	SPI_ERR: 0 = NO_ERR 1 = ERR
25	RW	0x0	ADMA_ERR: 0 = NO_ERR 1 = ERR
24	RW	0x0	AUTO_CMD12_ERR: 0 = NO_ERR 1 = ERR
23	RW	0x0	CURRENT_LIMIT_ERR: 0 = NO_ERR 1 = POWER_FAIL
22	RW	0x0	DATA_END_BIT_ERR: 0 = NO_ERR 1 = ERR
21	RW	0x0	DATA_CRC_ERR: 0 = NO_ERR 1 = ERR
20	RW	0x0	DATA_TIMEOUT_ERR: 0 = NO_ERR 1 = TIMEOUT

Bit	R/W	Reset	Description
19	RW	0x0	COMMAND_INDEX_ERR: 0 = NO_ERR 1 = ERR
18	RW	0x0	COMMAND_END_BIT_ERR: 0 = NO_ERR 1 = END_BIT_ERR_GENERATED
17	RW	0x0	COMMAND_CRC_ERR: 0 = NO_ERR 1 = CRC_ERR_GENERATED
16	RW	0x0	COMMAND_TIMEOUT_ERR: 0 = NO_ERR 1 = TIMEOUT
15	RO	X	ERR_INTERRUPT: 0 = NO_ERR 1 = ERR
8	RO	X	CARD_INTERRUPT: 0 = NO_INT 1 = GEN_INT
7	RW	0x0	CARD_REMOVAL: 0 = NO_INT 1 = GEN_INT
6	RW	0x0	CARD_INSERTION: 0 = NO_INT 1 = GEN_INT
5	RW	0x0	BUFFER_READ_READY: 0 = NO_INT 1 = GEN_INT
4	RW	0x0	BUFFER_WRITE_READY: 0 = NO_INT 1 = GEN_INT
3	RW	0x0	DMA_INTERRUPT: 0 = NO_INT 1 = GEN_INT
2	RW	0x0	BLOCK_GAP_EVENT: 0 = NO_INT 1 = GEN_INT
1	RW	0x0	XFER_COMPLETE: 0 = NO_INT 1 = GEN_INT
0	RW	0x0	CMD_COMPLETE: 0 = NO_INT 1 = GEN_INT

25.4.14 SDMMC_INTERRUPT_STATUS_ENABLE_0

This register is used to select which interrupt status is indicated to the Host System as the interrupt. These status bits all share the same1 bit interrupt line. Setting any of these bits to 1 enables interrupt generation.

Normal Interrupt Status Enable Register

Offset: 034h | Read/Write: R/W | Reset: 0b00000x0000000000xxxxxxx000000000

Bit	Reset	Description
31:30	0x0	VENDOR_SPECIFIC_ERR: 0 = DISABLE

Bit	Reset	Description
		3 = ENABLE
29	0x0	CEATA_ERROR: 0 = NO_ERROR 1 = ERROR
28	0x0	TARGET_RESP_ERROR: 0 = NO_ERROR 1 = ERROR
27	0x0	SPI_ERR: 0 = DISABLE 1 = ENABLE
25	0x0	ADMA_ERR: 0 = DISABLE 1 = ENABLE
24	0x0	AUTO_CMD12_ERR: 0 = DISABLE 1 = ENABLE
23	0x0	CURRENT_LIMIT_ERR: 0 = DISABLE 1 = ENABLE
22	0x0	DATA_END_BIT_ERR: 0 = DISABLE 1 = ENABLE
21	0x0	DATA_CRC_ERR: 0 = DISABLE 1 = ENABLE
20	0x0	DATA_TIMEOUT_ERR: 0 = DISABLE 1 = ENABLE
19	0x0	COMMAND_INDEX_ERR: 0 = DISABLE 1 = ENABLE
18	0x0	COMMAND_END_BIT_ERR: 0 = DISABLE 1 = ENABLE
17	0x0	COMMAND_CRC_ERR: 0 = DISABLE 1 = ENABLE
16	0x0	COMMAND_TIMEOUT_ERR: 0 = DISABLE 1 = ENABLE
8	0x0	CARD_INTERRUPT: 0 = DISABLE 1 = ENABLE
7	0x0	CARD_REMOVAL: 0 = DISABLE 1 = ENABLE
6	0x0	CARD_INSERTION: 0 = DISABLE 1 = ENABLE
5	0x0	BUFFER_READ_READY: 0 = DISABLE 1 = ENABLE
4	0x0	BUFFER_WRITE_READY: 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
3	0x0	DMA_INTERRUPT: 0 = DISABLE 1 = ENABLE
2	0x0	BLOCK_GAP_EVENT: 0 = DISABLE 1 = ENABLE
1	0x0	TRANSFER_COMPLETE: 0 = DISABLE 1 = ENABLE
0	0x0	COMMAND_COMPLETE: 0 = DISABLE 1 = ENABLE

25.4.15 SDMMC_INTERRUPT_SIGNAL_ENABLE_0

This register is used to select which interrupt status is notified to the Host System as the interrupt. These status bits all share the same 1 bit interrupt line. Setting any of these bits to 1 enables interrupt generation.

Normal Interrupt Signal Enable Register

Offset: 038h | Read/Write: R/W | Reset: 0b00000x0000000000xxxxxxx000000000

Bit	Reset	Description
31:30	0x0	VENDOR_SPECIFIC_ERR: 0 = DISABLE 3 = ENABLE
29	0x0	CEATA_ERROR: 0 = NO_ERROR 1 = ERROR
28	0x0	TARGET_RESP_ERROR: 0 = NO_ERROR 1 = ERROR
27	0x0	SPI_ERR: 0 = DISABLE 1 = ENABLE
25	0x0	ADMA_ERR: 0 = DISABLE 1 = ENABLE
24	0x0	AUTO_CMD12_ERR: 0 = DISABLE 1 = ENABLE
23	0x0	CURRENT_LIMIT_ERR: 0 = DISABLE 1 = ENABLE
22	0x0	DATA_END_BIT_ERR: 0 = DISABLE 1 = ENABLE
21	0x0	DATA_CRC_ERR: 0 = DISABLE 1 = ENABLE
20	0x0	DATA_TIMEOUT_ERR: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
19	0x0	COMMAND_INDEX_ERR: 0 = DISABLE 1 = ENABLE
18	0x0	COMMAND_END_BIT_ERR: 0 = DISABLE 1 = ENABLE
17	0x0	COMMAND_CRC_ERR: 0 = DISABLE 1 = ENABLE
16	0x0	COMMAND_TIMEOUT_ERR: 0 = DISABLE 1 = ENABLE
8	0x0	CARD_INTERRUPT: 0 = DISABLE 1 = ENABLE
7	0x0	CARD_REMOVAL: 0 = DISABLE 1 = ENABLE
6	0x0	CARD_INSERTION: 0 = DISABLE 1 = ENABLE
5	0x0	BUFFER_READ_READY: 0 = DISABLE 1 = ENABLE
4	0x0	BUFFER_WRITE_READY: 0 = DISABLE 1 = ENABLE
3	0x0	DMA_INTERRUPT: 0 = DISABLE 1 = ENABLE
2	0x0	BLOCK_GAP_EVENT: 0 = DISABLE 1 = ENABLE
1	0x0	TRANSFER_COMPLETE: 0 = DISABLE 1 = ENABLE
0	0x0	COMMAND_COMPLETE: 0 = DISABLE 1 = ENABLE

25.4.16 SDMMC_AUTO_CMD12_ERR_STATUS_0

COMMAND_NOT_ISSUED - Command Not Issued By Auto CMD12 Error

Setting this bit to 1 means CMD_wo_DAT is not executed due to an Auto CMD12 Error (D04-D01) in this register.

INDEX_ERR - Auto CMD12 Index Error

This bit is set if the Command Index error occurs in response to a command.

END_BIT_ERR - Auto CMD12 End Bit Error

This bit is set when detecting that the end bit of command response is 0.

CRC_ERR - Auto CMD12 CRC Error

This bit is set when detecting a CRC error in the command response.

TIMEOUT_ERR - Auto CMD12 Timeout Error

This bit is set if no response is returned within 64 SDCLK cycles from the end bit of command. If this bit is set to1, the other error status bits (D04-D02) are meaningless.

NOT_EXECUTED - Auto CMD12 Not Executed

If memory multiple block data transfer is not started due to command error, this bit is not set because it is not necessary to issue Auto CMD12. Setting this bit to 1 means the Host Controller cannot issue Auto CMD12 to stop memory multiple block data transfer due to some error. If this bit is set to 1, other error status bits (D04-D01) are meaningless.

The relation between Auto CMD12 CRC Error and Auto CMD12 Timeout Error is shown below

Auto CMD12 CRC Error	Auto CMD12 Timeout Error	Kinds of error
0	0	No Error
0	1	Response Timeout Error
1	0	Response CRC Error
1	1	CMD line conflict

Auto CMD12 Error Status Register

Offset: 03ch | Read/Write: RO | Reset: 0bxxxxxxxx

Bit	Reset	Description
7	X	COMMAND_NOT_ISSUED: 0 = NO_ERR 1 = NOT_ISSUED
4	X	INDEX_ERR: 0 = NO_ERR 1 = ERR
3	X	END_BIT_ERR: 0 = NO_ERR 1 = END_BIT_ERR_GENERATED
2	X	CRC_ERR: 0 = NO_ERR 1 = CRC_ERR_GENERATED
1	X	TIMEOUT_ERR: 0 = NO_ERR 1 = TIMEOUT
0	X	NOT_EXECUTED: 0 = EXECUTED 1 = NOT_EXECUTED

25.4.17 SDMMC_CAPABILITIES_0

SPI_BLOCK_MODE

Setting 1 indicates spi block mode is supported

SPI_MODE

Setting 1 indicates spi mode is supported

SYSTEM_BUS_64BIT_SUPPORT - 64-bit System Bus Support

Setting 1 to this bit indicates that the Host Controller supports 64-bit address descriptor mode and is connected to 64-bit address system bus.

INTERRUPT_MODE

Setting 1 indicates interrupt mode is supported

VOLTAGE_SUPPORT_1_8_V - Voltage Support 1.8V

VOLTAGE_SUPPORT_3_0_V - Voltage Support 3.0V

VOLTAGE_SUPPORT_3_3_V - Voltage Support 3.3V

SUSPEND_RESUME_SUPPORT - Suspend/Resume Support

This bit indicates whether the Host Controller supports Suspend / Resume functionality. If this bit is 0, the Host Driver shall not issue either Suspend or Resume commands because the Suspend and Resume mechanism is not supported.

DMA_SUPPORT - SDMA Support

This bit indicates whether the Host Controller is capable of using SDMA to transfer data between system memory and the Host Controller directly.

HIGH_SPEED_SUPPORT - High Speed Support

This bit indicates whether the Host Controller and the Host System support High Speed mode and they can supply SD Clock frequency from 25MHz to 50MHz.

ADMA1_SUPPORT - ADMA1 Support

This bit indicates whether the Host Controller is capable of using ADMA1.

ADMA2_SUPPORT - ADMA2 Support

This bit indicates whether the Host Controller is capable of using ADMA2.

EXTENDED_MEDIA_BUS_SUPPORT

Setting to 1, indicates 8-bit data bus is supported.

MAX_BLOCK_LENGTH - Max Block Length

This value indicates the maximum block size that the Host Driver can read and write to the buffer in the Host Controller. The buffer shall transfer this block size without wait cycles. Three sizes can be defined as indicated below. It is noted that transfer block length shall be always 512 bytes for SD Memory Cards regardless this field.

BASE_CLOCK_FREQUENCY - Base Clock Frequency for SD Clock

This value indicates the base (maximum) clock frequency for the SD Clock. Unit values are 1MHz. If the real frequency is 16.5MHz, the lager value shall be set 01 0001b (17MHz) because the Host Driver use this value to calculate the clock divider value (Refer to the SDCLK Frequency Select in the Clock Control register.) and it shall not exceed upper limit of the SD Clock frequency. The supported clock range is 10MHz to 63MHz. If these bits are all 0, the Host System has to get information via another method.

- Not 0 - 1MHz to 63MHz
- 000000b- Get information via another method

TIMEOUT_CLOCK_UNIT - Timeout Clock Unit

This bit shows the unit of base clock frequency used to detect Data Timeout Error.

- 0 KHz
- 1 MHz

TIMEOUT_CLOCK_FREQUENCY - Timeout Clock Frequency

This bit shows the base clock frequency used to detect Data Timeout Error. The Timeout Clock Unit defines the unit of this field's value.

- Timeout Clock Unit =0 [KHz] unit: 1KHz to 63KHz
- Timeout Clock Unit =1 [MHz] unit: 1MHz to 63MHz
- Not 0 - 1KHz to 63KHz or 1MHz to 63MHz
- 000000b - Get information via another method

Capabilities Register

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

Bit	Reset	Description
30	X	SPI_BLOCK_MODE: 0 = NOT_SUPPORTED 1 = SUPPORTED
29	X	SPI_MODE: 0 = NOT_SUPPORTED 1 = SUPPORTED
28	X	SYSTEM_BUS_64BIT_SUPPORT: 0 = NOT_SUPPORTED 1 = SUPPORTED
27	X	INTERRUPT_MODE: 0 = NOT_SUPPORTED 1 = SUPPORTED
26	X	VOLTAGE_SUPPORT_1_8_V: 0 = NOT_SUPPORTED 1 = SUPPORTED
25	X	VOLTAGE_SUPPORT_3_0_V: 0 = NOT_SUPPORTED 1 = SUPPORTED
24	X	VOLTAGE_SUPPORT_3_3_V: 0 = NOT_SUPPORTED 1 = SUPPORTED
23	X	SUSPEND_RESUME_SUPPORT: 0 = NOT_SUPPORTED 1 = SUPPORTED
22	X	DMA_SUPPORT: 0 = NOT_SUPPORTED 1 = SUPPORTED
21	X	HIGH_SPEED_SUPPORT: 0 = NOT_SUPPORTED 1 = SUPPORTED
20	X	ADMA1_SUPPORT: 0 = NOT_SUPPORTED 1 = SUPPORTED
19	X	ADMA2_SUPPORT: 0 = NOT_SUPPORTED 1 = SUPPORTED
18	X	EXTENDED_MEDIA_BUS_SUPPORT: 0 = NOT_SUPPORTED 1 = SUPPORTED
17:16	X	MAX_BLOCK_LENGTH: 0 = BYTE512

Bit	Reset	Description
		1 = BYTE1024 2 = BYTE2048 3 = RESERVED
13:8	X	BASE_CLOCK_FREQUENCY
7	X	TIMEOUT_CLOCK_UNIT: 0 = KHZ 1 = MHZ
5:0	X	TIMEOUT_CLOCK_FREQUENCY

25.4.18 SDMMC_MAXIMUM_CURRENT_0

Maximum Current Capabilities Register

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

Bit	Reset	Description
23:16	X	MAXIMUM_CURRENT_FOR_1_8V: Maximum Current for 1.8V
15:8	X	MAXIMUM_CURRENT_FOR_3_0V: Maximum Current for 3.0V
7:0	X	MAXIMUM_CURRENT_FOR_3_3V: Maximum Current for 3.3V

25.4.19 SDMMC_FORCE_EVENT_0

The Force Event Register is not a physically implemented register. Rather, it is an address at which the Auto CMD12 Error Status Register can be written.

Writing 1 : set each bit of the Auto CMD12 Error Status Register

Writing 0 : no effect

Force Event for Auto CMD12 Error Status Register

Offset: 050h | Read/Write: R/W | Reset: 0b00000x0000000000xxxxxxxx0xx00000

Bit	Reset	Description
31:30	0x0	VENDOR_SPECIFIC_ERR_STATUS: 0 = DISABLE 3 = ENABLE
29	0x0	CEATA_ERROR: 0 = NO_ERROR 1 = ERROR
28	0x0	TARGET_RESP_ERROR: 0 = NO_ERROR 1 = ERROR
27	0x0	SPI_ERR: 0 = DISABLE 1 = ENABLE
25	0x0	ADMA_ERR: 0 = NO_INTERRUPT 1 = INTERRUPT
24	0x0	AUTOCMD12_ERR: 0 = NO_INTERRUPT 1 = INTERRUPT
23	0x0	CURRENTLIMIT_ERR: 0 = NO_INTERRUPT

Bit	Reset	Description
		1 = INTERRUPT
22	0x0	DATA_END_BIT_ERR: 0 = NO_INTERRUPT 1 = INTERRUPT
21	0x0	DATACRC_ERR: 0 = NO_INTERRUPT 1 = INTERRUPT
20	0x0	DATATIMEOUT_ERR: 0 = NO_INTERRUPT 1 = INTERRUPT
19	0x0	COMMAND_INDEX_ERR: 0 = NO_INTERRUPT 1 = INTERRUPT
18	0x0	COMMAND_END_BIT_ERR: 0 = NO_INTERRUPT 1 = INTERRUPT
17	0x0	COMMAND_CRC_ERR: 0 = NO_INTERRUPT 1 = INTERRUPT
16	0x0	COMMAND_TIMEOUT_ERR: 0 = NO_INTERRUPT 1 = INTERRUPT
7	0x0	AUTO_CMD12_NOT_ISSUED: 0 = NO_INTERRUPT 1 = INTERRUPT
4	0x0	AUTO_CMD12_INDEX_ERR: 0 = NO_INTERRUPT 1 = INTERRUPT
3	0x0	AUTO_CMD12_END_BIT_ERR: 0 = NO_INTERRUPT 1 = INTERRUPT
2	0x0	AUTO_CMD12_CRC_ERR: 0 = NO_INTERRUPT 1 = INTERRUPT
1	0x0	AUTO_CMD12_TIMEOUT_ERR: 0 = NO_INTERRUPT 1 = INTERRUPT
0	0x0	AUTO_CMD12_NOT_EXECUTED: 0 = NO_INTERRUPT 1 = INTERRUPT

25.4.20 SDMMC_ADMA_ERR_STATUS_0

ADMA_LENGTH_MISMATCH_ERR - ADMA Length Mismatch Error. This error occurs in the following 2 cases.

- While Block Count Enable being set, the total data length specified by the Descriptor table is different from that specified by the Block Count and Block Length.
- Total data length cannot be divided by the block length.

ADMA_ERR_STATE - ADMA Error State. This field indicates the state of ADMA when error is occurred during ADMA data transfer. This field never indicates "10" because ADMA never stops in this state.

D01 - D00	ADMA Error State when error is occurred	Contents of SYS_SDR register
00	ST_STOP (Stop DMA)	Points next of the error descriptor

D01 - D00	ADMA Error State when error is occurred	Contents of SYS_SDR register
01	ST_FDS (Fetch Descriptor)	Points the error descriptor
10	Never set this state	(Not used)
11	ST_TFR (Transfer Data)	Points the next of the error descriptor

ADMA Error Status Register

Offset: 054h | Read/Write: R/W | Reset: 0b000

Bit	Reset	Description
2	0x0	ADMA_LENGTH_MISMATCH_ERR: 0 = NO_ERR 1 = ERR
1:0	0x0	ADMA_ERR_STATE

25.4.21 SDMMC_ADMA_SYSTEM_ADDRESS_0

This register holds byte address of executing command of the Descriptor table. 32-bit Address Descriptor uses lower 32-bit of this register. At the start of ADMA, the Host Driver shall set start address of the Descriptor table. The ADMA increments this register address, which points to next line, when every fetching a Descriptor line. When the ADMA Error Interrupt is generated, this register shall hold valid Descriptor address depending on the ADMA state. The Host Driver programs Descriptor Table on 32-bit boundary and set 32-bit boundary address to this register. ADMA2 ignores lower 2-bit of this register and assumes it to be 00b.

ADMA System Address Register

Offset: 058h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	ADMA_SYSTEM_ADDRESS

25.4.22 SDMMC_SPI_INTERRUPT_SUPPORT_0

SPI Interrupt Support Register

Offset: 0f0h | Read/Write: RO | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	SPI_INT_SUPPORT: This bit is set to indicate the assertion of interrupts in SPI MODE at any time Irrespective on the status of card select.

25.4.23 SDMMC_SLOT_INTERRUPT_STATUS_0

- VENDOR_VERSION_NUMBER - Vendor Version Number

  This status is reserved for the vendor version number. The Host Driver should not use this status.
- SPECIFICATION_VERSION_NUMBER - Specification Version Number

  This status indicates the Host Controller Spec. Version. The upper and lower 4-bits indicate the version.
  - 00 SD Host Specification Version 1.00
  - 01 SD Host Specification Version 2.00

  Including the feature of the ADMA and Test Register, others Reserved
- INTERRUPT_SIGNAL_FOR_EACH_SLOT - Interrupt Signal For Each Slot

These status bits indicate the logical OR of Interrupt Signal and Wakeup Signal for each slot. A maximum of 8 slots can be defined. If one interrupt signal is associated with multiple slots, the Host Driver can know which interrupt is generated by reading these status bits. By a power on reset or by setting Software Reset For All, the interrupt signal shall be de-asserted and this status shall read 00h.

- Bit 00 Slot 1
- Bit 01 Slot 2
- Bit 02 Slot 3
- Bit 07 Slot 8

Slot Interrupt Status Register

Offset: 0fch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:24	X	VENDOR_VERSION_NUMBER
23:16	X	SPECIFICATION_VERSION_NUMBER
7:0	X	INTERRUPT_SIGNAL_FOR_EACH_SLOT

25.5 Vendor Specific SDMMC Registers

The following Registers are Vendor Specific Registers and are mapped to Vendor Specific Address Space (0x100 - 0x1FF)

25.5.1 SDMMC_VENDOR_CLOCK_CNTRL_0

Vendor Clock Control Register

Offset: 100h | Read/Write: R/W | Reset: 0b1

Bit	Reset	Description
0	0x1	SDMMC_CLK: This is set when sdmmc_clk is supplied by the CAR module.Prior to sdmmc_clk switch OFF. This bit should be written '0'. Prior to sdmmc_clk switch OFF. This bit should be written '0'. By writing zero, the asynchronous card interrupt is routed to the Interrupt controller. 0 = DISABLE 1 = ENABLE

25.5.2 SDMMC_VENDOR_SPI_CNTRL_0

Vendor SPI Control Register

Offset: 104h | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
0	0x0	SPI_MODE: 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 de-assert the CS Signal 0 = DISABLE 1 = ENABLE

25.5.3 SDMMC_VENDOR_SPI_INTR_STATUS_0

The fields are valid when an SPI error has occurred.

SPI Interrupt Status Register

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

Bit	Reset	Description
8:5	X	DAT_ERR_TOKEN: Data Error Token, while read from card.
4:0	X	DAT_RESPONSE: Data Response while write to card 5 = DATA_ACCEPTED 11 = CRC_ERR 13 = WRITE_ERR

25.5.4 SDMMC_VENDOR_CEATA_CNTRL_0

Vendor CEATA Control Register

Offset: 10ch | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
0	0x0	CCS_SIGNAL: If this bit is set to 1,the controller expects a Command completion signal from the card after the transfer. If the CCS Signal doesn't come within Data Timeout Value the CEATA Error is flagged. 0 = DISABLE 1 = ENABLE

25.5.5 SDMMC_VENDOR_BOOT_CNTRL_0

This Register is used to configure Boot Mode to support MMC v4.3 cards.

Vendor Boot Control Register

Offset: 110h | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
1	0x0	BOOT_ACK: This bit is used to support Boot Option in MMC 4.3 version cards. If set Boot acknowledgment is given by card else not given by card 0 = DISABLE 1 = ENABLE
0	0x0	BOOT: This bit enables/disable BootOption1.If set BootOption1 is enable, HW auto clears it when boot data is done. Writing 0 terminates the BootOption1 0 = DISABLE 1 = ENABLE

25.5.6 SDMMC_VENDOR_BOOT_ACK_TIMEOUT_0

Vendor Boot Acknowledgment Timeout Register

Offset: 114h | Read/Write: R/W | Reset: 0b00000000000000000000

Bit	Reset	Description
19:0	0x0	VALUE: If Boot Acknowledgment is not received within the programmed number of cycles. Boot Acknowledgement Timeout error occurs(VENDOR_SPECIFIC_ERR[0])

25.5.7 SDMMC_VENDOR_BOOT_DAT_TIMEOUT_0

Boot Data Timeout Register

Offset: 118h | Read/Write: R/W | Reset: 0b00000000000000000000000000

Bit	Reset	Description
24:0	0x0	VALUE: If Boot Data is not received within the programmed number of cycles. Then Data Timeout error occurs.

25.5.8 SDMMC_VENDOR_DEBOUNCE_COUNT_0

The Debounce Counter runs on 32KHz clock.

Keeping the default value to 100ms = (100 * 32cycles/1ms) = 3200 cycles for 100ms = 0xC80

Debounce Counter Value Register

Offset: 11ch | Read/Write: R/W | Reset: 0b00000000000000110010000000

Bit	Reset	Description
23:0	0xc80	VALUE: The number of 32KHz clock cycles is programmed to meet Debounce period of the card slot.

26.0 USB COMPLEX

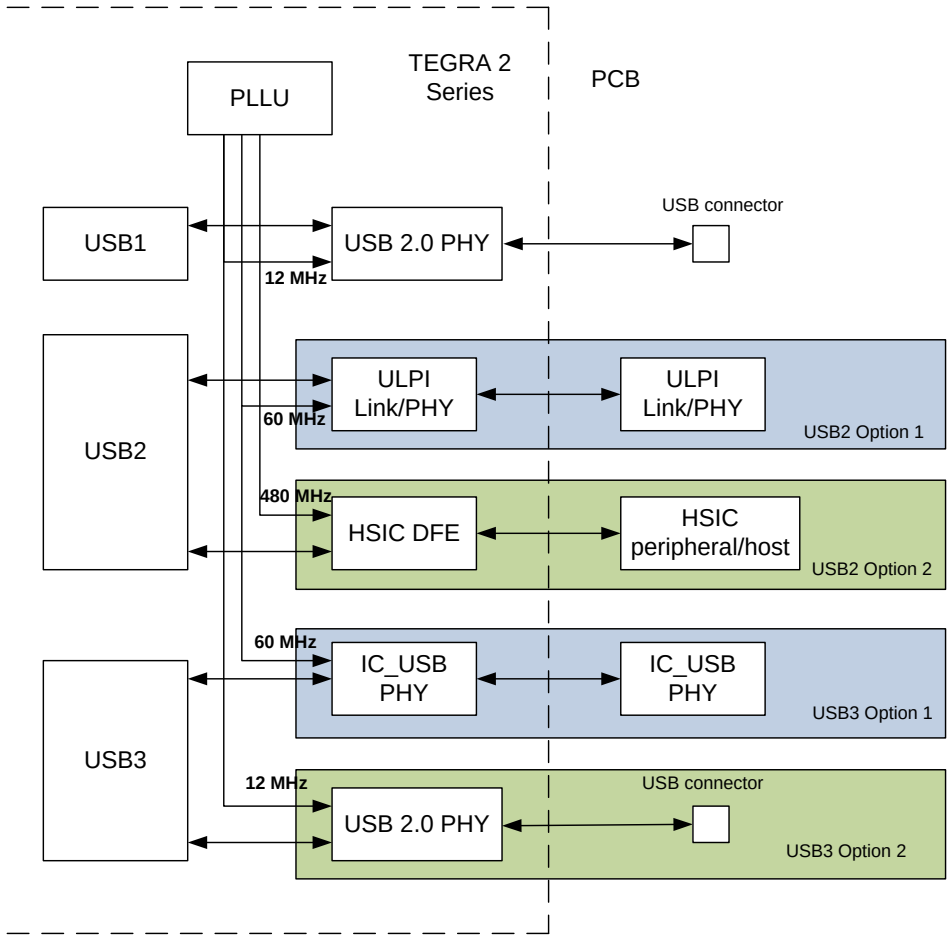
The USB complex in Tegra® 2 Series provides a mechanism to communicate with a PC and/or a peripheral such as a USB keyboard, mouse, camera, etc. using regular USB ports and to an on-board baseband controller using either USB/ULPI/HSIC/ICUSB interfaces. It consists of 3 USB controllers and 6 USB interfaces: UTMIP1, Link ULPI, Null ULPI, HSIC, ICUSB and UTMIP3. Figure 68 shows the relationship between different USB controllers and interfaces.

Tegra 2 Series devices implement VBUS and ID sensors required for device/host functionality and associated interrupt mechanisms; software uses these for the device/host state machine in software. Battery charger detection is implemented as specified in USB Battery charger specification version 1.0; software needs to implement the charger sequence.

The USB controllers implement transaction level processing of control requests. Software decodes the type of control request and programs the packets appropriately. Suspend/resume functionality saves power at times when USB functionality is not required; there are automatic wake-up events and interrupts that software can program.

26.1 USB Controllers and Interfaces

Figure 68. USB Controllers and Interfaces



USB Device/Host and UTMIP1 (USB1 port)

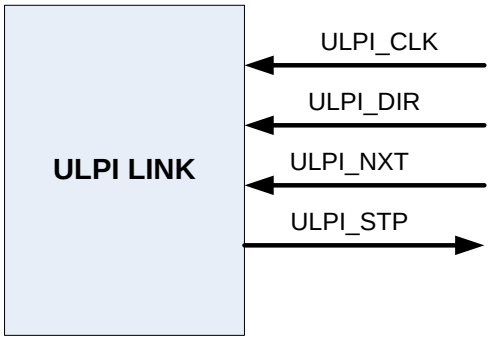
This is the primary USB port on Tegra 2 Series devices. It supports USB device and USB host modes. USB recovery is supported on this interface only. Wake-up from deep sleep mode is also supported on VBUS and accessory detect (ACCx_DETECT) pins. Power and Ground pins for USB1 and USB3 are shared.

USB2 and Link ULPI

This interface supports connecting an on-board external ULPI PHY to use as additional USB port. Depending on ULPI PHY capabilities, it can support USB Host and USB Peripheral modes of operation.

Link ULPI interface shares the pins with Null ULPI and other interfaces of Tegra 2 Series devices. It is required that the pin-mux be programmed prior to using the Link ULPI interface.

Link ULPI is a slave on the ULPI bus, in that the 60 MHz ULPI interface clock is driven from the external ULPI PHY on the ULPI_CLK pin. Note the change of direction in the pins ULPI_DIR, ULPI_NXT and ULPI_STP with respect to Null ULPI interface.

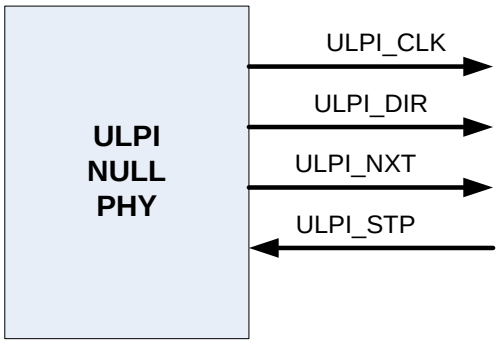


USB2 and Null ULPI

This interface allows Tegra 2 Series devices to behave like a PHY so that an external Link ULPI controller can be connected to it. This is used to support baseband controllers that use ULPI interface.

Null ULPI interface shares the pins with Link ULPI and other interfaces of Tegra 2 Series devices. It is required that the pin-mux be programmed prior to using the Null ULPI interface.

Null ULPI is a master on the ULPI bus, in that the 60 MHz ULPI interface clock is driven by Tegra 2 Series devices to the external Link on the ULPI_CLK pin. Note the change of direction in the pins ULPI_DIR, ULPI_NXT and ULPI_STP with respect to Link ULPI interface.



USB2 and UHSIC

This interface allows connection of an on-board HSIC Peripheral/Host to the Tegra 2 Series devices. This is used to support baseband controllers that use HSIC interface.

UHSIC supports both Host and Peripheral modes. It does not support switching between Host and Peripheral modes. Since the external HSIC Peripheral/Host is always on-board, the role of Peripheral and Host is expected to be fixed at board design time.

USB3 and ICUSB

This interface allows connection of an on-board ICUSB Peripheral to the Tegra 2 Series devices. This is used to support baseband controllers that use ICUSB interface.

ICUSB interface on Tegra 2 Series supports two power-supply voltages: 1.8V and 3.0V. Software need to perform proper power sequencing as described in the ICUSB specification.

USB3 and UTMIP3 (USB3 port)

This is the second USB port on Tegra 2 Series devices. It supports Host and Peripheral modes of operation. This could be also called the Host port, as the primary purpose of this interface is to support Host mode.

Power and Ground pins for USB1 and USB3 are shared.

26.1.1 Interface Restrictions

Link ULPI, Null ULPI and UHSIC are all exclusive interfaces on USB2 controller, and only one of the interfaces can be used in a given board design. Similarly, ICUSB and UTMIP3 are exclusive interfaces on USB3 controller, only one of the interfaces can be used in a given board design.

On power-on-reset, USB2 defaults to Link ULPI mode, and USB3 defaults to ICUSB mode.

26.2 Controller

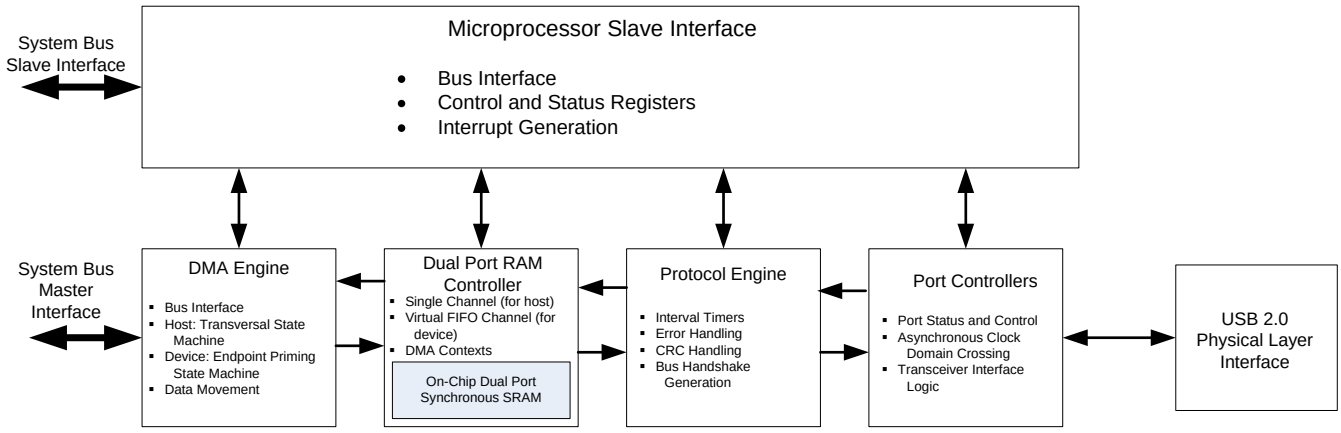
All three USB controllers offer the same functionality, although the interface may restrict any one to a subset of the possible functions.

The USB Controller can be either a USB host or device. It has a USB 2.0 High Speed dual role USB Host controller or USB Device controller.

- Device controller registers and data structures are implemented as extensions to the EHCI programmers interface.
- Supports USB legacy (USB 1.1) Full and Low speed devices without a companion USB 1.1 host controller or host controller driver software using EHCI standard data structures.

The block diagram of USB is shown in the figure below.

Figure 69. USB Controller Block Diagram



26.2.1 Endpoint Capabilities

All controllers support 16 bi-directional endpoints in device mode. Endpoint 0 is always a bi-directional control endpoint. All other endpoints can be programmed as one of Bulk, Interrupt, Isochronous or Control type in either direction. Control endpoints are always bi-directional and hence to program an endpoint as control type, both IN and OUT directions need to be programmed to control type. Conversely, if an endpoint has either IN or OUT direction programmed to be non-control type, the other direction also needs to be programmed to be non-control type.

The max packet size supported on any endpoint is 512 bytes in high-speed mode, for both device and host modes.

26.2.2 AHB Interface

All USB controllers use an AHB interface as master and slave for communicating with other parts of Tegra 2 Series devices. Table 82 lists the AHB interface numbers for each controller.

Table 82: AHB Master and Slave Numbers

Controller	AHB Master Number	AHB Slave Number
USB1	6	6
USB2	18	9
USB3	17	11

26.3 USB Programming Guidelines

26.3.1 USB Device/Host Programming

Clock Initialization

After power-on-reset, the USB clocks are disabled. Software can enable the USB clocks by setting CLK_ENB_USBD in CLK_OUT_ENB_L register to ENABLE.

Also, USB stays in reset and software should bring it out of reset by first setting SWR_USBD_RST in RST_DEVICES_L register to ENABLE and then set it to DISABLE.

There are some parameters in UTMIP that need to be programmed before bringing the UTMIP out of reset. Those need to be programmed every time USB is reset. That includes SWR_USBD_RST in RST_DEVICES_L register and RST in USB2D_USBCMD register.

After UTMIP is reset, PHY clock takes some time to come up, so software must wait until PCLKVLD bit in MISC_USB_OTG register becomes valid. This bit, if set to 1, also generates an interrupt if RSM_IE is set in MISC_USB_OTG register.

26.3.1.1 ID and VBUS Connection

With Tegra 2 Series devices, you can use ACCx_DETECT supported by the UTMIP/USB hardware, or use a GPIO for ID functions. If using a GPIO as the ID, the GPIO can be sampled to detect the presence of a micro-A or micro-B plug and the sampled value can be written to SW_ID bit in MISC_USB_OTG register. It is recommended that board design connects ACCx_DETECT pin on the Tegra 2 Series devices to the ID pin on the micro-AB connector even if the ID is connected to a GPIO.

Tegra 2 Series devices have a VBUS pin connected to UTMIP and VBUS pin from the connector should be connected to this pin even though VBUS from the connector is connected to a GPIO. It is not required that VBUS from the connector is directly connected to Tegra 2 Series' VBUS pin. It is required that its voltage is above 3V when VBUS is on.

Both VBUS and ACCx_DETECT are available as a wake-up event during DPD/LP0 state. Please consult PMC programming guidelines for further recommendations.

26.3.1.2 Detection of USB Cable Insertion

The USB cable insertion event can be detected by a change in the ACCx_DETECT 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 2 Series devices because of the weak pull-up on this pin. This makes the Tegra 2 Series devices start-up as a B-device, even though a cable is not connected (When the ID pin is seen as a 1 by Tegra 2 Series devices, there can be no-cable or micro-B connected).

When a user attaches the USB cable, one of the following two events occur:

- User connects micro-A end of the cable to Tegra 2 Series devices. This changes the ID pin to 0. Hence the software can detect the ID change and go to A-device mode. Software can then supply power to the USB and places the Tegra 2 Series devices in host mode.
- User connects micro-B (or mini-B) end of the cable to Tegra 2 Series devices. The other end of the cable can be either micro-A or standard-A and will be connected to a standard host, respectively. For both cases, the Host will start driving power on VBUS and software can detect the cable insertion event by checking the voltage on VBUS has gone above A_Sess_Vld level by reading the A_VBUS_VLD_STS bit in USB_PHY_VBUS_SENSORS register. Once this is seen, it can place the Tegra 2 Series devices in the device mode.

Note: ID change can be detected by enabling the GPIO interrupt.
VBUS change can be detected by enabling the interrupt A_VBUS_VLD_INT_EN in USB_PHY_VBUS_SENSORS register.
USB controller and PHY clocks need not be turned on for this detection.
To detect these events during DPD/LP0 modes, use VBUS/ID wakeup events in PMC.

26.3.1.3 PHY Clock Control

The PHY clock can be turned off using the bit SUSP_SET in MISC_USB_OTG 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 MISC_USB_OTG register. This bit must be pulsed by first writing a 1 and then writing a 0 to the bit.

The current suspend status of the PHY can be checked by reading the bit SUSPENDED in MISC_USB_OTG register. If the PHY is placed in suspend, then this bit will be set to 1, else it will be set to 0. Also, if this bit is 1, PHY clock will be turned off and PCLKVLD in MISC_USB_OTG register will be set to 0. If this bit is 0, PCLKVLD bit can indicate whether the PHY clock is turned on or not.

The PHY clock can be turned on automatically on the events mentioned below. Set only the required wakeup enable bits based on the state of the USB as mentioned below. Don't set unnecessary wakeup enable bits. When the PHY clock wakes up under any such event, make sure that the wakeup enable bits are turned off as soon as possible.

There is an interrupt generated whenever PHY clock is turned on which can be checked by checking the value of PCLKVLD in MISC_USB_OTG register to be 1. The interrupt can be enabled/disabled by setting the bit RSM_IE in MISC_USB_OTG to 1/0.

Note:	The PLLU_ENABLE bit in PLLU_BASE register should be always set to ENABLE. All the parameters for PLLU in PLLU_BASE and PLLU_MISC register should be set according to the oscillator frequency used. Not doing that will put the USB PHY PLL in a free-running mode, and can result in undesirable side-effects. When the USB PHY is suspended, the PCLK should not be stopped. It can be reduced to 32 KHz, however. If the system clock is stopped, then it may not be possible to bring up the system on a wakeup event described below.

There are two cases to consider for controlling the PHY clocks.

Device Mode

The PHY clock can be turned off in two cases:

- When USB cable is not connected
- When the USB Host puts the USB bus in suspend mode

When USB cable isn't connected, the VBUS signal is 0. In this case, the software needs to turn on the PHY clock on cable insertion. Follow the procedure described above in the section on Detection of cable insertion.

When USB cable is connected, the PHY clock can only be turned off when the bit SLI in USB2D_USBSTS register is set to 1 when the USB Host puts the USB bus in suspend mode. This bit can be used to generate an interrupt if the bit SLE in USB2D_USBINTR register if set to 1. In this case, software needs to set two bits in the MISC_USB_OTG 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 MISC_USB_OTG register is set to 1, the PHY clock can be turned on automatically on receiving USB resume event from USB Host. If WAKE_ON_DISCON_EN in MISC_USB_OTG register is set to 1, the PHY clock can be turned on automatically on receiving USB reset event from USB Host or if the USB cable is disconnected. These bits should only be turned on after the PHY clock is turned off. The method described above can be used to interrupt the processor when the PHY clock is turned on. Also, turn off these bits when coming out of suspend.

Host Mode

When a device is not connected, software can set the WKCN bit in USB2D_PORTSC1 register. Make sure that VBUS is also turned on. After this, it can stop the PHY clock by using the method described above. The PHY clock will resume automatically when a device is connected. The method described above can be used to interrupt the processor when the PHY clock is turned on.

With a device connected, software needs to put the USB system under suspend by setting the bit SUSP in USB2D_PORTSC1 register. It needs to wait until this bit is set to 1 by the controller to make sure that the USB system actually went into suspend. It can enable the bits WK_RSM_EN in MISC_USB_OTG register and WK_DS in USB2D_PORTSC1 register. Then it can stop the PHY clock as described above. The PHY clock will be turned on automatically if the USB device disconnects or it does a remote wakeup signaling. There will be an interrupt associated with this as described above.

If the software wants to wakeup USB system, then it needs to turn on the PHY clock as described above.

26.3.1.4 Initialization and Shutdown Procedure for USB Device/Host (Legacy Mode)

1. Program UTMIP and PLLU parameters.
   - Set register UTMIP_MISC_CFG0 (0x7000:0a24) UTMIP_SUSPEND_EXIT_ON_EDGE (bit 22) to 0.

- Set the PLLU_ENABLE to 1, and never ever de-assert it.

2. Bring up USB clocks by writing 1 to CLK_OUT_ENB_L (0x6000:6010) USBD (bit 22).
3. Assert and de-assert RST_DEVICES_L (0x6000:6004) USBD (bit 22) to bring the USB block out of reset.
   - Wait until PHY_CLKVALID is set to 1.
4. Set USB controller and PHY to reset by writing 1 to USBD(bit 22) in RST_DEVICES_L register (0x6000:6004). This will have the same effect as putting USB PHY to suspend, and hence the power consumption should be low.
   - Wait until PHY_CLKVALID is set to 0.
5. Set all PD bits required to bring USB pads to low power mode.
   - UTMIP_OTGPD (bit 11) in UTMIP_BIAS_CFG0 (0x7000:0a0c) is needed for VBUS detection, and so it should be set to 0.
   - If using VBUS_WAKEUP for VBUS detection, then UTMIP_BIASPD (bit 10) in UTMIP_BIAS_CFG0 (0x7000:0a0c) 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 (bit 12) in MISC_USB_OTG (0x7000:0028) should be set to 1 to enable ID detection.
   - UTMIP_IDPD_SEL (bit 22) in UTMIP_BIAS_CFG0 (0x7000:0a0c) should be set to 0.
6. When (VBUS=1) or (ID=0) is detected, then do the following:
   - Bring USB controller and PHY out of reset by writing 0 to USBD (bit 1) in RST_DEVICES_L register (0x6000:6004).
   - Clear the PD bits that are required for the normal operation.
7. From this on, driver can run the normal operations.
   - UTMIP should only be suspended by generating a positive pulse on SUSP_SET (bit 16) in MISC_USB_OTG (0x7000:0028) register when USB cable is connected and only after USB bus is in suspend, for both device and host modes.
8. If a cable disconnect is detected (VBUS = 0) or (ID = 1), then go back to step 4.

26.3.1.5 Charger Detection 1 (Non-compliant Chargers)

There can be 4 conditions for the different kind of non-compliant chargers: SE1, FS-PU, LS-PU, SE0. There can be some hooks in SW so that any kind of non-compliant charger can be supported by following the generic procedure below:

1. VBUS detection (by whatever means SW does it, doesn't matter)
2. Enable USB controller and PHY power and clocks
3. Wait until PHY_CLKVALID = 1.
4. Wait for 10 microseconds more. This gives time for any filtering to pass through so SW doesn't read wrong values.
5. Read the LS bits [11:10] in PORTSC register.
   - Case 1: LS = 2'b11: Connected to charger_1.
   - Case 2: LS = 2'b01: Connected to charger_2.
   - Case 3: LS = 2'b10: Connected to charger_3.
   - Case 4: LS = 2'b00: Connected to either charger_4 or a PC host.
6. If case 1, 2, or 3, enable charging current as per that charger requirement. This could be different for every implementation.
7. If case 4, either connected to PC host or a charger_4 (which could be USB compliant charger as well). Go to step 10.
8. Set the USB circuit on Tegra 2 Series devices to low-power mode now. Make sure the charger circuit doesn't change state.
9. VBUS disconnect. Go back to step 1.

10. Check if you are connected to a USB compliant charger by doing the charger detection from VBAT register. If yes, go back to step 8. If no, go to next step.
11. Try to set USB controller to device mode and enable by setting it to RUN mode. Wait for SOF.
12. If you don't see SOF after timeout, then you are connected to charger_4. Set current limit as required for that charger. Go to Step 7.

26.3.1.6 Charger Detection 2 (Compliant Chargers)

This substitutes step 10 from previous section.

1. Make sure UTMIP_PD_CHRG in APB_MISC_UTMIP_BAT_CHRG_CFG0 register is set to 0, so that charger detection circuit is on.
2. Set the other register bits in APB_MISC_UTMIP_BAT_CHRG_CFG0 register to the following values:
   - UTMIP_OP_SRC_EN = 1
   - UTMIP_ON_SINK_EN = 1
   - UTMIP_OP_SINK_EN = 0
   - UTMIP_ON_SRC_EN = 0
3. Wait for about 10 microseconds to allow the battery charger detector to settle.
4. Now check the following bits in the register USB_PHY_VBUS_WAKEUP_ID.
   - VDAT_DET_CHG_DET bit is set, and VDAT_DET_STS indicates 1. Then we are connected to a USB compliant charger.
   - If it isn't set, then we are connected to a PC host.
5. Set the registers bits in APB_MISC_UTMIP_BAT_CHRG_CFG0 register to the following values:
   - UTMIP_OP_SRC_EN = 0
   - UTMIP_ON_SINK_EN = 0
   - UTMIP_OP_SINK_EN = 0
   - UTMIP_ON_SRC_EN = 0
6. Now normal USB operation can continue if we are connected to PC host.

26.3.2 USB1 Device/Host Programming Non-Legacy Mode

USB1 supports another programming mode, the non-legacy mode. Legacy mode behaves the same way as Tegra 2 devices where a few registers for USB1 are located under APB_MISC base address. In non-legacy mode, all these registers are accessible under the USB base address.

Non-legacy mode can be selected by setting the bit USB1_NO_LEGACY_MODE in USB1_LEGACY_CTRL register. This has to be done before doing any other accesses to the USB1 and UTMIP1 after resetting the USB1 controller.

The following sections show the changes required during non-legacy mode.

26.3.2.1 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 and software should bring it out of reset by first setting SWR_USBD_RST in RST_DEVICES_L register to ENABLE and then set it to DISABLE.

At this point software can set the bit USB1_NO_LEGACY_MODE in USB1_LEGACY_CTRL register to switch to non-legacy mode. Please set the bit UTMIP_RESET in USB_SUSP_CTRL register before changing the legacy mode setting to keep the UTMIP1 in reset.

There are some parameters in UTMIP that need to be programmed before bringing the UTMIP out of reset. Those need to be programmed every time USB is reset. That includes SWR_USBD_RST in RST_DEVICES_L register, RST in USB2D_USBCMD register and UTMIP_RESET in USB_SUSP_CTRL register.

After UTMIP is reset, PHY clock takes some time to come up, so software must wait until USB_PHY_CLKVALID bit in USB_SUSP_CTRL register becomes valid. This bit, if set to 1, also generates an interrupt if USB_PHY_CLK_VALID_INT_ENB is set in USB_SUSP_CTRL register. The interrupt status can be read at USB_PHY_CLK_VALID_INT_STS bit in USB_SUSP_CTRL register.

26.3.2.2 ID and VBUS Connection

With Tegra 2 Series devices, we can use ACCx_DETECT supported by the UTMIP/USB hardware, or use a GPIO for ID functions. If using a GPIO as a ID, the GPIO can be sampled to detect the presence of a micro-A or micro-B plug and the sampled value can be written to ID_SW_VALUE bit in USB_PHY_VBUS_WAKEUP_ID register. It is recommended that board design connects ACCx_DETECT on Tegra 2 Series devices to the ID pin on the micro-AB connector even if ID is connected to a GPIO.

Tegra 2 Series devices have a VBUS pin connected to UTMIP and VBUS pin from the connector should be connected to this pin even though VBUS from the connector is connected to a GPIO. It is not required that VBUS from the connector is directly connected to Tegra 2 Series devices' 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. Please consult PMC programming guidelines for this.

26.3.2.3 Detection of USB Cable Insertion

The USB cable insertion event can be detected by a change in the ID or VBUS pin on the micro-AB connector. Initially, when the cable is not attached, the ID pin on the micro-AB connector is floating and is detected as 1 by the Tegra 2 Series devices because of the weak pull-up on this pin. This makes the Tegra 2 Series devices start-up as a B-device, even though a cable is not connected (When the ID pin is seen as a 1 by Tegra 2 Series devices, there can be no-cable or micro-B connected).

When a user attaches the USB cable, one of the following two events occur:

- User connects micro-A end of the cable to Tegra 2 Series devices. This changes the ID pin to 0. Hence the software can detect the ID change and go to A-device mode. Software can then supply power to the USB and places the Tegra 2 Series devices in host mode.
- User connects micro-B (or mini-B) end of the cable to Tegra 2 Series devices. The other end of the cable can be either micro-A or standard-A and will be connected to a standard host, respectively.

For both cases, the Host will start driving power on VBUS and software can detect the cable insertion event by checking the voltage on VBUS has gone above A_Sess_Vld level by reading the A_VBUS_VLD_STS bit in USB_PHY_VBUS_SENSORS register. Once this is seen, it can place the Tegra 2 Series devices in the device mode.

Note:
- ID change can be detected by enabling the GPIO interrupt.
- VBUS change can be detected by enabling the interrupt A_VBUS_VLD_INT_EN in USB_PHY_VBUS_SENSORS register.
- USB controller clock needs to be turned on for this detection. However, PHY can be kept in reset/suspend.
- To detect these events during DPD/LP0 modes, use VBUS/ID wakeup events in PMC.

26.3.2.4 PHY Clock Control

The PHY clock can be turned off using the bit SUSP_SET in USB_SUSP_CTRL register. This bit must be pulsed by first writing a 1 and then writing a 0 to the bit.

To turn on the PHY clock, software should write to SUSP_CLR in USB_SUSP_CTRL register. This bit must be pulsed by first writing a 1 and then writing a 0 to the bit.

The current suspend status of the PHY 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.

The PHY clock can be turned on automatically on the events mentioned below. Set only the required wakeup enable bits based on the state of the USB as mentioned below. Don't set unnecessary wakeup enable bits. When the PHY clock wakes up under any such event, make sure that the wakeup enable bits are turned off as soon as possible.

To prevent the PHY clock from waking up on unnecessary glitches on the USB pins, software can set the USB_WAKEUP_DEBOUNCE_COUNT field in USB_SUSP_CTRL register to a non-zero value (between 1-7). This will allow the wakeup event to be debounced by the equivalent number of HCLK cycles.

There is an interrupt generated whenever PHY is waked up from suspend which can be checked by checking the value of USB_WAKEUP_INT_STS in USB_SUSP_CTRL register to be 1. The interrupt can be enabled/disabled by setting the bit USB_WAKEUP_INT_ENB in USB_SUSP_CTRL to 1/0.

There is an interrupt generated whenever PHY clock is turned on which can be checked by checking the value of USB_PHY_CLK_VALID_INT_STS in USB_SUSP_CTRL register to be 1. The interrupt can be enabled/disabled by setting the bit USB_PHY_CLK_VALID_INT_ENB in USB_SUSP_CTRL to 1/0.

To clear the interrupts USB_WAKEUP_INT_STS and USB_PHY_CLK_VALID_INT_STS, software can write a 1 to corresponding bits.

Generally, whenever PHY is waked up from suspend, first wakeup event is generated (USB_WAKEUP_INT_STS = 1) followed by PHY clock valid interrupt (USB_PHY_CLK_VALID_INT_STS = 1) when PHY clock starts up.

Note:

- The PLLU_ENABLE bit in PLLU_BASE register should be always set to ENABLE. All the parameters for PLLU in PLLU_BASE and PLLU_MISC register should be set according to the oscillator frequency used. Not doing that will put the USB PHY PLL in a free-running mode, and can result in undesirable side-effects.
- When the USB PHY is suspended, the USB1 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 USB1 is in suspend, AHB clocks to USB1 controller is turned off, and hence there should be no register access to USB2_CONTROLLER_* registers. All registers marked as ARUSB1_IF_* are accessible, and can be used to resume the UTMIP1 PHY clocks.

There are two cases to consider for controlling the PHY clocks:

Device Mode

The PHY clock can be turned off in two cases:

- When USB cable isn't connected
- when the USB Host puts the USB bus in suspend mode

When USB cable isn't connected, the VBUS signal is 0. In this case, the software needs to turn on the PHY clock on cable insertion. Follow the procedure described above in the section on Detection of cable insertion.

When USB cable is connected, the PHY clock can only be turned off when the bit SLI in USB2D_USBSTS register is set to 1 when the USB Host puts the USB bus in suspend mode. This bit can be used to generate an interrupt if the bit SLE in USB2D_USBINTR register if set to 1.

In this case, software needs to set two bits in the USB_SUSP_CTRL register to turn on the PHY clock on a USB resume or a USB reset event from USB Host: WAKE_ON_DISCON_EN_DEV: Wake on Disconnect enable during device mode, and USB_WAKE_ON_RESUME_EN - Wake on Resume enable.

If USB_WAKE_ON_RESUME_EN bit in USB_SUSP_CTRL register is set to 1, the PHY clock can be turned on automatically on receiving USB resume event from USB Host. If WAKE_ON_DISCON_EN_DEV bit in USB_SUSP_CTRL register is set to 1, the PHY clock can be turned on automatically on receiving USB reset event from USB Host or if the USB cable is disconnected. These bits should only be turned on after the PHY clock is turned off. The method described above can be used to interrupt the processor when the PHY clock is turned on. Also, turn off these bits when coming out of suspend.

Host Mode

When a device is not connected, software can set the WKCN bit in USB2D_PORTSC1 register. Make sure that VBUS is also turned on. After this, it can stop the PHY clock by using the method described above. The PHY clock will resume automatically when a device is connected. The method described above can be used to interrupt the processor when the PHY clock is turned on.

With a device connected, software needs to put the USB system under suspend by setting the bit SUSP in USB2D_PORTSC1 register. It needs to wait until this bit is set to 1 by the controller to make sure that the USB system actually went into suspend. It can enable the bits USB_WAKE_ON_RESUME_EN in USB_SUSP_CTRL register and WK_DS in USB2D_PORTSC1 register. Then it can stop the PHY clock as described above.

The PHY clock will be turned on automatically if the USB device disconnects or it does a remote wakeup signaling. There will be an interrupt associated with this as described above.

If the software wants to wakeup USB system, then it needs to turn on the PHY clock as described above.

26.3.2.5 Initialization and Shutdown Procedure for USB

1. Bring up USB clocks by writing 1 to CLK_ENB_USBD in CLK_OUT_ENB_L register.
2. Assert and de-assert SWR_USBD_RST in RST_DEVICES_L register to bring USB1 out of reset.
3. Set UTMIP_RESET bit in USB_SUSP_CTRL register to 1 to keep UTMIP1 PHY in reset.
4. Set USB1_NO_LEGACY_MODE bit in USB1_LEGACY_CTRL register to 1 to switch to non-legacy mode.
5. Program UTMIP and PLLU parameters.
   - Set register UTMIP_MISC_CFG0 (0x7000:0a24) UTMIP_SUSPEND_EXIT_ON_EDGE (bit 22) to 0.
   - Set the PLLU_ENABLE to 1, and never ever de-assert it.
6. Set UTMIP_RESET bit in USB_SUSP_CTRL register to 0 to bring UTMIP1 out of reset.
   - Wait until PHY_CLKVALID is set to 1.
7. Set USB controller and PHY to suspend by writing 1 to SUSP_SET bit in USB_SUSP_CTRL register. This will reduce the power consumption on USB1 controller and UTMIP1 PHY.
   - Wait until PHY_CLKVALID is set to 0.
8. 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.
9. To interrupt the processor on VBUS or ID detection, software can set the following:

- Enable appropriate VBUS interrupt enable, for example, I using A_SESS_VLD sensor for VBUS detection, set A_SESS_VLD_INT_EN in USB_PHY_VBUS_SENSORS register to 1.
- Enable ID detection interrupt by setting ID_INT_EN in USB_PHY_VBUS_WAKEUP_ID register.

10. When (VBUS=1) or (ID=0) is detected, do the following:
    - Bring USB controller and PHY out of suspend by pulsing SUSP_CLR in USB_SUSP_CTRL register by first writing 1 and then writing 0.
        - Wait until PHY_CLKVALID is set to 1.
    - Clear the PD bits that are required for the normal operation.
11. From here on, driver can run the normal operations.
    - UTMIP should only be suspended by generating a positive pulse on SUSP_SET in USB_SUSP_CTRL register when USB cable is connected and only after USB bus is in suspend, for both device and host modes.
12. If a cable disconnect is detected (VBUS = 0) or (ID = 1), then go back to step 4.

26.3.3 USB2 Programming Sequence

USB2 controller supports Link ULPI, Null ULPI and UHSIC interfaces. All of these interfaces are directly controlled through USB2 register space.

26.3.3.1 Clock Initialization

After power-on-reset, USB2 clocks are disabled. Software can enable USB2 clocks by setting CLK_ENB_USB2 in ARCLK_RST_CLK_OUT_ENB_H register to ENABLE.

Also, USB2 stays in reset at power-on and software should bring it out of reset by first setting SWR_USB2_RST in RST_DEVICES_H register to ENABLE and then set it to DISABLE.

After the clocks for USB2 are up, SW can program any registers for USB2 required for proper configuration.

PLLU outputs a 60 MHz clock FO_ICUSB for Null ULPI interface and a 480 MHz clock FO_UHSIC for UHSIC interface. If using any of these interfaces, PLLU should be programmed appropriately.

Any time after USB2 is reset, or ULPI/UHSIC PHY comes out of suspend, software needs to wait until ULPI/UHSIC PHY clock comes up. It can do that by checking for USB_PHY_CLK_VALID bit in USB2_IF_USB_SUSP_CTRL register. If this bit is 1, PHY clocks are up, else they aren't. There are interrupts associated with this bit: if USB_PHY_CLK_VALID_INT_ENB is set to ENABLE, USB_PHY_CLK_VALID_INT_STS will be set to SET whenever PHY clock becomes valid. Software can write a 1 to USB_PHY_CLK_VALID_INT_STS to clear the interrupt status.

Before doing anything useful on USB2, one of its interfaces needs to be selected and configured.

26.3.3.2 Selection of Link ULPI interface

If using Link or Null ULPI interface, software can hold UHSIC in reset by setting UHSIC_RESET in USB2_IF_USB_SUSP_CTRL register to 1.

1. Disable pull-ups and pull-downs on ULPI pins by setting UAA_PU_PD and UAB_PU_PD fields in APB_MISC_PP_PULLUPDOWN_REG_D and UDA_PU_PD in APB_MISC_PP_PULLUPDOWN_REG_E to NORMAL (2’b00).
2. Enable ULPI interface by writing 1 to ULPI_PHY_ENB bit in USB2_IF_USB_SUSP_CTRL register.
3. Bypass the pin-mux on ULPI outputs by writing 1 to ULPI_OUTPUT_PINMUX_BYP and ULPI_CLKOUT_PINMUX_BYP fields in USB2_IF_ULPI_TIMING_CTRL_0 register.
4. Remove the tri-state on ULPI pins by setting UAA, UAB fields in APB_MISC_PP_TRISTATE_REG_B register and UDA field in APB_MISC_PP_TRISTATE_REG_D register to NORMAL (1’b0).

26.3.3.3 Selection of Null ULPI interface

If using Link or Null ULPI interface, software can hold UHSIC in reset by setting UHSIC_RESET in USB2_IF_USB_SUSP_CTRL register to 1.

1. Program the PLLU parameters.
2. Disable pull-ups and pull-downs on ULPI pins by setting UAA_PU_PD and UAB_PU_PD fields in APB_MISC_PP_PULLUPDOWN_REG_D and UDA_PU_PD in APB_MISC_PP_PULLUPDOWN_REG_E to NORMAL (2'b00).
3. Enable ULPI interface by writing 1 to ULPI_PHY_ENB bit in USB2_IF_USB_SUSP_CTRL register.
4. Set the ULPI pin-mux settings by writing 1 to the fields ULPI_OUTPUT_PINMUX_BYP, ULPI_CLKOUT_PINMUX_BYP, ULPI_CLK_OUT_ENA, ULPI_CLK_PADOUT_ENA, ULPI_SHADOW_CLK_SEL, and ULPI_CORE_CLK_SEL in USB2_IF_ULPI_TIMING_CTRL_0 register.
5. Enable PLLU 60 MHz output by writing 1 to ULPIS2S_PLLU_MASTER_BLASTER60 field in USB2_IF_USB_ULPIS2S_CTRL register.
6. Remove the tri-state on ULPI pins by setting UAA, UAB fields in APB_MISC_PP_TRISTATE_REG_B register and UDA field in APB_MISC_PP_TRISTATE_REG_D register to NORMAL (1'b0).

26.3.3.4 Selection of UHSIC interface

UHSIC IO pad is in power-down state at power-on. Bring it out of power-down mode by writing PD_BG, PD_TX, PD_TRK, PD_RX, PD_ZI and RPD_DATA, RPD_STROBE fields in UHSIC_PADS_CFG1 register to 0.

1. Hold UHSIC In reset by writing 1 to UHSIC_RESET field in USB2_IF_USB_SUSP_CTRL register.
2. Program the PLLU parameters to enable UHSIC PLLU clock.
3. Select UHSIC interface by writing 1 to UHSIC_PHY_ENB field in USB2_IF_USB_SUSP_CTRL register.
4. Program the configuration parameters for UHSIC.
5. Release reset to UHSIC by writing 0 to UHSIC_RESET field in USB2_IF_USB_SUSP_CTRL register.
6. Program the USB2 controller to use UHSIC PHY by writing 0 to PTS field in USB2_CONTROLLER_USB2D_PORTSC1 register.
7. Program the UHSIC_HS_POSTAMBLE_OUTPUT_ENABLE field in the UHSIC_TX_CFG0 register to "1".
8. Wait until PHY clock comes up by checking for USB_PHY_CLK_VALID bit in USB2_IF_USB_SUSP_CTRL register. Interrupt can be generated as explained above.

There is no VBUS or ID detection required for this interface as these pins don't exist for UHSIC.

26.3.3.5 ID and VBUS detection in Link ULPI mode

For external ULPI PHY, the ID and VBUS can be connected to the USB connector as per the guidelines given by ULPI PHY vendor. The detection for ID and VBUS can be done in two ways:

- USB2 controller registers USB2_CONTROLLER_USB2D_OTGSC can be used to enable ID_PU (ID pull-up), and read status on VBUS and ID sensors.
- Use the USB2D_ULPI_VIEWPORT register for USB2 to perform writes/reads to the external ULPI PHY.

Setting the PP bit in USB2D_PORTSC1 register would enable the VBUS going out to the USB connector when USB2 is configured in Host mode. The board designer needs to connect appropriate signals from the ULPI PHY in order to make correct use of this feature.

Alternately, software can write appropriate register in ULPI PHY according to the datasheet of the particular PHY using the USB2D_ULPI_VIEWPORT register.

Note that USB2 is disabled during DPD (LP0) mode. If system designer wants to wake up the system on a cable connection for USB2, then ID and VBUS should be brought to PMU and PMU can be programmed to wake up the Tegra 2 Series devices on an event on these pins. Alternately, these can be driven to GPIOs and those GPIOs can be used in PMC as wakeup sources.

For details on USB2D_ULPI_VIEWPORT, please see the description for that register in the register section.

26.3.3.6 Detection of USB Cable Insertion in Link ULPI mode

The USB cable insertion event can be detected by a change in the ID or VBUS pin on the micro-AB connector connected to ULPI PHY. Initially, when the cable is not attached, the ID pin on the micro-AB connector is floating and is detected as 1 by Tegra 2 Series devices because of the weak pull-up on this pin. This makes the Tegra 2 Series devices USB2 start-up as a B-device, even though a cable is not connected (When the ID pin is seen as a 1 by Tegra 2 Series devices, there can be no-cable or micro-B connected).

When a user attaches the USB cable to the ULPI PHY port, one of the following two events occur:

- User connects micro-A end of the cable to Tegra 2 Series devices. This changes the ID pin to 0. Hence the software can detect the ID change by reading the USB2D_OTGSC register and go to A-device mode. Software can then supply power to VBUS on ULPI PHY port and place the Tegra 2 Series device USB2 in host mode.
- User connects micro-B (or mini-B) end of the cable to Tegra 2 Series devices. The other end of the cable can be either micro-A or standard-A, and will be connected to a standard host, respectively.

For both cases, the Host will start driving power on VBUS and software can detect the cable insertion event by checking the voltage on VBUS has gone above A_Vbus_Vld level using the USB2D_OTGSC register. Once this is seen, it can place the Tegra 2 Series device USB2 in the device mode.

26.3.3.7 PHY Clock control

The PHY clock can be turned off by writing 1 to the bit PHCD in USB2D_PORTSC1 register.

> **Note:** There is no need to pulse this bit as opposed to USB1 controller.

To turn on the PHY clock, software should write to USB_SUSP_CLR in USB2_IF_USB_SUSP_CTRL register. This bit must be pulsed by first writing a 1 and then writing a 0 to the bit.

Whenever the PHY is placed in suspend, PHY clock will be turned off and USB_PHY_CLK_VALID in USB2_IF_USB_SUSP_CTRL register will be set to 0. Software should make sure that PHY clock shuts down by polling this bit whenever it places the PHY into suspend. Software should make sure that PHY clock shuts down by polling this bit whenever it places the PHY into suspend.

The PHY clock can be turned on automatically on the events mentioned below. Set only the required wakeup enable bits based on the state of the USB as mentioned below. Don't set unnecessary wakeup enable bits. When the PHY clock wakes up under any such event, make sure that the wakeup enable bits are turned off as soon as possible.

To prevent the PHY clock from waking up on unnecessary glitches on the USB pins, software can set the USB_WAKEUP_DEBOUNCE_COUNT field in USB_SUSP_CTRL register to a non-zero value (between 1-7). This will allow the wakeup event to be debounced by the equivalent number of HCLK cycles.

There is an interrupt generated whenever PHY is waked up from suspend which can be checked by checking the value of USB_WAKEUP_INT_STS in USB_SUSP_CTRL register to be 1. The interrupt can be enabled/disabled by setting the bit USB_WAKEUP_INT_ENB in USB_SUSP_CTRL to 1/0.

There is an interrupt generated whenever PHY clock is turned on which can be checked by checking the value of USB_PHY_CLK_VALID_INT_STS in USB_SUSP_CTRL register to be 1. The interrupt can be enabled/disabled by setting the bit USB_PHY_CLK_VALID_INT_ENB in USB_SUSP_CTRL to 1/0.

To clear the interrupts USB_WAKEUP_INT_STS and USB_PHY_CLK_VALID_INT_STS, software can write a 1 to corresponding bits.

Generally, whenever PHY is waked up from suspend, first wakeup event is generated (USB_WAKEUP_INT_STS = 1) followed by PHY clock valid interrupt (USB_PHY_CLK_VALID_INT_STS = 1) when PHY clock starts up.

Note:

- This suspend is not supported in null ULPI mode.
- When doing this suspend for link ULPI mode,

a. USB2 clock should be enabled (CLK_ENB_USBD in ARCLK_RST_CLK_OUT_ENB_L register should be set to ENABLE).

b. USB2 reset should be disabled (SWR_USB2_RST in RST_DEVICES_H register should be set to DISABLE).

Not doing this could lead to undesirable side-effects.

- When USB2 is in suspend, AHB clocks to USB2 controller is turned off, and hence there should be no register access to USB2_CONTROLLER_1_* registers. All registers marked as ARUSB2_IF_* are accessible, and can be used to resume the ULPI PHY clocks.

There are two cases to consider for controlling the ULPI PHY clocks.

Device Mode

The PHY clock can be turned off in two cases:

- When USB cable isn't connected
- When the USB Host puts the USB bus in suspend mode

When USB cable isn't connected, the VBUS signal is 0. In this case, the software needs to turn on the PHY clock on cable insertion. Follow the procedure described above in the section on Detection of cable insertion.

When USB cable is connected, the PHY clock can only be turned off when the bit SLI in USB2D_USBSTS register is set to 1 when the USB Host puts the USB bus in suspend mode. This bit can be used to generate an interrupt if the bit SLE in USB2D_USBINTR register if set to 1. In this case, software needs to set two bits in the USB2_IF_USB_SUSP_CTRL register to turn on the PHY clock on a USB resume or a USB reset event from USB Host: USB_WAKE_ON_DISCON_EN_DEV: Wake on Disconnect enable in device mode, and USB_WAKE_ON_RESUME_EN - Wake on Resume enable.

If USB_WAKE_ON_RESUME_EN in USB2_IF_USB_SUSP_CTRL register is set to 1, the PHY clock can be turned on automatically on receiving USB resume event from USB Host. If USB_WAKE_ON_DISCON_EN_DEV in USB2_IF_USB_SUSP_CTRL register is set to 1, the PHY clock can be turned on automatically on receiving USB reset event from USB Host or if the USB cable is disconnected. These bits should only be turned on after the PHY clock is turned off. The method described above can be used to interrupt the processor when the PHY clock is turned on.

Also, turn off these bits when coming out of suspend.

Host Mode

When a device is not connected, software can set the WKCN bit in USB2D_PORTSC1 register. Make sure that VBUS is also turned on. After this, it can stop the PHY clock by using the method described above. The PHY clock will resume automatically when a device is connected. The method described above can be used to interrupt the processor when the PHY clock is turned on.

With a device connected, software needs to put the USB system under suspend by setting the bit SUSP in USB2D_PORTSC1 register. It needs to wait until this bit is set to 1 by the controller to make sure that the USB system actually went into suspend. It can enable the bits USB_WAKE_ON_RESUME_EN in USB2_IF_USB_SUSP_CTRL register and WK_DS in USB2D_PORTSC1 register. Then it can stop the PHY clock as described above. The PHY clock will be turned on

automatically if the USB device disconnects or it does a remote wakeup signaling. There will be an interrupt associated with this as described above.

If the software wants to wakeup USB system, then it needs to turn on the PHY clock as described above.

26.3.3.8 Bus Signaling Procedure changes for UHSIC interface

To support the UHSIC interface, some changes are required in the bus signaling procedure for USB2 controller (connect and bus reset sequences). These are described in the sections below:

Host Mode - Bus Connect Sequence

The normal connect sequence assumes that port power has been enabled on a downstream port and the device has signaled a “Connect” on the bus. This will generate a port change interrupt and the software takes action from here:

1. Check if “LineState” (field LS in USB2D_PORTSC1 register) is DPlus – Make sure the HSIC bus is IDLE before we start the connect;
2. Program the XCVR to detect SHORT connect by writing “1” to DETECT_SHORT_CONNECT field in the UHSIC_MISC_CFG0 register.
3. Program the FORCE_XCVR_MODE field in UHSIC_MISC_CFG0 register to “1”.
4. Program the PTC field in the PORTSC1 register to “FORCE_ENABLE” .
5. Poll until “Device Connect” is received by polling until the UHSIC_CONNECT_DETECT field in the UHSIC_STAT_CFG0 register becomes “1”.
6. Set PTC to “TEST_MODE_DISABLE” - Exit the test mode and let USB2 controller reach state “HS Connected Idle”. There's no need to wait between steps 2 and 3;
7. Poll until “LineState” is SE0 - Wait until USB2 controller acknowledges a HSIC IDLE Bus;
8. Poll until “CurrentConnectStatus” (field CCS in USB2D_PORTSC1 register) is enabled – This guarantees the port has connected successfully before proceeding;
9. Check “PortSpeed” (field PSPD in USB2D_PORTSC1 register) – The ultimate check is to verify the bus connection speed as reported by the USB2 controller. It should match the HSIC native speed (High Speed).

Host Mode: Bus Reset Sequence

This is not the standard connect but rather a re-connect procedure without physically detaching the USB cable. This operation is demanded by software whenever it is necessary as it is the case when an unrecoverable anomaly takes place. It would then bring the bus, the port and the device into a well-known state.

1. Check if PortReset (PR bit in USB2D_PORTSC1 register) is NOT active – If by any means a bus reset is already taking place it must be stopped before proceeding. This is done by clearing the bit and polling until it goes LOW;
2. Enable the “PortReset” bit - Start the HSIC bus reset on a downstream port by activating the PR bit of the USB2D_PORTSC1 register;
3. Wait until the end of the reset – This wait time is defined by USB 2.0 spec and has the effect of holding the reset signaling on the HSIC bus;
4. Clear the PR bit – Stop the reset signaling by clearing the respective port reset bit;
5. Poll until the reset bit goes LOW – It is advised to guarantee that the reset bit is effectively cleared;
6. Poll until LineState is DPlus – This confirms that the reset signaling has really finished on the HSIC bus and that the USB2 controller is ready to connect again;
7. Proceed with step 2 of the “HOST Bus Connect Sequence” as described above. From here on, the procedure is the same as with an initial connect, instructing the USB2 controller to enter HS directly, skipping speed negotiation.

Device Mode: Bus Connect Sequence

Generally (assuming all initial procures have already been run), a connect will occur when both the RUNSTOP bit in USB2D_USBCMD register has been enabled and the Host has enabled the port power on its side. There is no physical connection event when dealing with a HSIC bus and this signaling mechanism guarantees a successful connect whichever component reaches the respective connection point first.

In a standard connection, software would wait for a “bus reset” System Interrupt to start the speed negotiation procedure. With a HSIC bus instead, reset does not take place and software must drive the USB2 controller into the “HS Connected Idle” state:

1. Enable the RUNSTOP bit in register USB2D_USBCMD – The HSIC PHY will detect this and wait for a bus IDLE meaning the bus has connected. When that happens, it will show LineState as DPlus and place the bus in the CONNECT state for at least two strobe periods;
2. Poll LineState until it reaches DPlus – This is needed because the Host may not be ready when the device starts the connect. We can only proceed to the “HS Connected Idle” state when the HOST has seen the connect;
3. Follow the exact instructions from step 3 of the Host “Bus Connect Sequence”.
4. Enable the Port Force Full Speed Connect (field PFSC in USB2D_PORTSC1 register) – This is necessary if system implementation is to account for bus resets during operation. Mostly this will be the case therefore the feature is mandatory. Otherwise a bus reset would drive the device USB2 controller into speed negotiation which is not allowed for an HSIC bus.

Device Mode: Bus Reset Sequence

A bus reset is always started by a Host. When this happens the USB2 controller Device will detect the reset and issue a system interrupt. Both the fields PR in USB2D_PORTSC1 register and URI in USB2D_USBSTS register are set and after the typical setup operations, in order to connect, software must also do:

1. Check if LineState is SE0 – If the Host has actually started to drive reset signaling on the bus, then the LineState should be SE0, otherwise the reset condition is false;
2. Poll until LineState is DPlus – This will make software execution synchronized with the end of the bus reset. After the bus reset and because of the “Force Full Speed” feature, the USB2 controller will be in “Full Speed Idle” state;
3. Follow step 3 of the Device “Bus Connect Sequence” – The procedure to follow is the same as above. Software must also enable the “Force Full Speed” feature at the end (as with step 4).

> **Note:** During the above described port connection procedures, due to the direct driving of the USB2 controller through a few of its internal states, some hardware interrupts will be skipped (this is why some steps involve polling cycles). Because of this, and depending on the programmer's choice, software may simulate/ignore some system interrupts (making interrupt routines to be called/attended from within the same execution thread) to maintain compatibility with the rest of the stack. This may apply for example to port change interrupts at the end of the bus reset for both Host/Device modes.

26.3.4 USB3 Programming Sequence

USB3 controller supports ICUSB and UTMIP3 interfaces. All of these interfaces are directly controlled through USB3 register space.

26.3.4.1 Clock Initialization

After power-on-reset, USB3 clocks are disabled. Software can enable USB3 clocks by setting CLK_ENB_USB3 in ARCLK_RST_CLK_OUT_ENB_H register to ENABLE.

Also, USB3 stays in reset at power-on and software should bring it out of reset by first setting SWR_USB3_RST in RST_DEVICES_H register to ENABLE and then set it to DISABLE.

After the clocks for USB3 are up, SW can program any registers for USB3 required for proper configuration.

PLLU outputs a 60 MHz clock FO_ICUSB for ICUSB interface and a 12 MHz clock FO_USB for UTMIP3 interface. If using any of these interfaces, PLLU should be programmed appropriately.

After anytime USB3 is reset, or ICUSB/UTMIP3 PHY comes out of suspend, software needs to wait until ICUSB/UTMIP3 PHY clock comes up. It can do that by checking for USB_PHY_CLK_VALID bit in USB3_IF_USB_SUSP_CTRL register. If this bit is 1, PHY clocks are up, else they aren't. There are interrupts associated with this bit: if USB_PHY_CLK_VALID_INT_ENB is set to ENABLE, USB_PHY_CLK_VALID_INT_STS will be set to SET whenever PHY clock becomes valid. Software can write a 1 to USB_PHY_CLK_VALID_INT_STS to clear the interrupt status.

Before doing anything useful on USB2, one of its interfaces needs to be selected and configured.

26.3.4.2 Selection of ICUSB Interface

If using ICUSB interface, software can hold UTMIP3 in reset by setting UTMIP3_RESET in USB3_IF_USB_SUSP_CTRL register to 1.

ICUSB interface only supports host mode of operation – a peripheral can be connected to Tegra 2 Series device's ICUSB port. Peripheral mode of operation isn't supported on ICUSB interface.

ICUSB IO pad is in power-down state at power-on. Bring it out of power-down mode by writing PD_ZI, PD_DR, and PD_TX fields in ICUSB_XCVR_CFG register to 0.

1. Program the PLLU parameters to enable ICUSB PLLU clock.
2. Select ICUSB interface by writing 1 to ICUSB_PHY_ENB field in USB3_IF_USB_SUSP_CTRL register.
3. Enable ICUSB module clock by writing 1 to ICUSB_MOD_CLK_ENB field in USB3_IF_USB_SUSP_CTRL register.
4. Wait until PHY clock comes up by checking for USB_PHY_CLK_VALID bit in USB2_IF_USB_SUSP_CTRL register. Interrupt can be generated as explained above.
5. Set CM field in USB2D_USBMODE register to HOST mode.
6. Change USB3 controller to ICUSB mode by writing iCUSB_SER to PTS field in USB2D_PORTSC register.
7. Enable appropriate voltage for ICUSB_VDD (1.8V or 3.0V) from PMU. Consult with the PMU vendor/datasheet about the programming required for this.
8. Enable ICUSB port in USB3 controller by writing 1 to IC_ENB1 field and appropriate ICUSB_VDD voltage in IC_VDD1 field of USB2D_ICUSB_CTRL register. For ICUSB_VDD of 1.8V, program 4 to IC_VDD1 and for ICUSB_VDD of 3.0V, program 5 to IC_VDD1. The value programmed in IC_VDD1 must match the voltage programmed in PMU for ICUSB_VDD.

Once initialization is done, standard USB drivers can be used to support ICUSB interface.

26.3.4.3 Selection of UTMIP3 interface

1. UTMIP3 IO pads are in power-down state at power-on. Bring them out of power-down mode by writing the following fields to 0:
   - FORCE_PD_POWERDOWN, FORCE_PD2_POWERDOWN, FORCE_PDZI_POWERDOWN fields in UTMIP_XCVR_CFG0 register.
   - OTGOD and BIASPD fields in UTMIP_BIAS_CFG0 register.
   - FORCE_PDDISC_POWERDOWEN , FORCE_PDCHRP_POWERDOWN, FORCE_PDDR_POWERDOWN field in UTMIP_XCVR_CFG1 register.
2. Hold UTMIP3 PHY in reset by writing UTMIP_RESET bit in USB3_IF_USB_SUSP_CTRL register to 1.
3. Program the PLLU parameters to enable UTMIP3 PLLU clock.
4. Enable UTMIP3 interface by setting UTMIP_PHY_ENB in USB3_IF_USB_SUSP_CTRL register to 1.
5. Program the configuration parameters for UTMIP3.

6. Release reset to UTMIP3 by writing 0 to UTMIP_RESET field in USB3_IF_USB_SUSP_CTRL register.
7. Wait until PHY clock comes up by checking for USB_PHY_CLK_VALID bit in USB3_IF_USB_SUSP_CTRL register. Interrupt can be generated as explained above.
8. Program the USB3 controller to use UTMIP3 PHY by setting the PTS field in USB2_CONTROLLER_USB2D_PORTSC1 register to UTMIP (2'b00).

26.3.4.4 ID and VBUS Connection

UTMIP3 is primarily meant to support additional USB Host functionality on Tegra 2 Series devices. Hence for UTMIP3, ID and VBUS pins aren't supported. GPIOs could be used in applications that need these pins. In such cases, software would set appropriate *_SW_VALUE and *_SW_EN bits for each VBUS and ID sensors depending on the value detected from the GPIOs.

Both VBUS and ID are available as a wake-up event during DPD/LP0 state. Please consult PMC programming guidelines for this.

26.3.4.5 Detection of USB Cable Insertion

The USB cable insertion event can be detected by a change in the ID or VBUS pin on the micro-AB connector. Initially, when the cable is not attached, the ID pin on the micro-AB connector is floating and is detected as 1 by Tegra 2 Series devices because of the weak pull-up on this pin. This makes Tegra 2 Series devices start-up as a B-device, even though a cable is not connected (When the ID pin is seen as a 1 by Tegra 2 Series devices, there can be no-cable or micro-B connected).

When a user attaches the USB cable, one of the following two events occur:

- User connects micro-A end of the cable to Tegra 2 Series devices. This changes the ID pin to 0. Hence the software can detect the ID change and go to A-device mode. Software can then supply power to the USB and place the Tegra 2 Series device in host mode.
- User connects micro-B (or mini-B) end of the cable to Tegra 2 Series devices. The other end of the cable can be either micro-A or standard-A and will be connected to a standard host, respectively.

For both cases, the Host will start driving power on VBUS and software can detect the cable insertion event by checking the VBUS status on corresponding GPIO. Once this is seen, it can place the Tegra 2 Series devices in the device mode.

Note: Both ID and VBUS change can be detected by enabling the corresponding GPIO interrupts.

USB controller/PHY clocks do not need to be turned on for this detection as this is done using GPIOs.

To detect these events during DPD/LP0 modes, use VBUS/ID wakeup events in PMC.

26.3.4.6 PHY Clock Control

The PHY clock can be turned off by writing 1 to the bit PHCD in USB2D_PORTSC1 register.

Note: There is no need to pulse this bit as opposed to USB1 controller.

To turn on the PHY clock, software should write to SUSP_CLR in USB3_IF_USB_SUSP_CTRL register. This bit must be pulsed by first writing a 1 and then writing a 0 to the bit.

The current suspend status of the PHY and thereby whether the PHY clock is running can be checked by reading the bit USB_PHY_CLK_VALID in USB_SUSP_CTRL register. If the PHY is placed in suspend, then this bit will be set to 0, else it will be set to 1. Software should make sure that PHY clock shuts down by polling this bit whenever it places the PHY into suspend.

The PHY clock can be turned on automatically on the events mentioned below. Set only the required wakeup enable bits based on the state of the USB as mentioned below. Don't set unnecessary wakeup enable bits. When the PHY clock wakes up under any such event, make sure that the wakeup enable bits are turned off as soon as possible.

To prevent the PHY clock from waking up on unnecessary glitches on the USB pins, software can set the USB_WAKEUP_DEBOUNCE_COUNT field in USB_SUSP_CTRL register to a non-zero value (between 1-7). This will allow the wakeup event to be debounced by the equivalent number of HCLK cycles.

There is an interrupt generated whenever PHY is waked up from suspend which can be checked by checking the value of USB_WAKEUP_INT_STS in USB_SUSP_CTRL register to be 1. The interrupt can be enabled/disabled by setting the bit USB_WAKEUP_INT_ENB in USB_SUSP_CTRL to 1/0.

There is an interrupt generated whenever PHY clock is turned on which can be checked by checking the value of USB_PHY_CLK_VALID_INT_STS in USB_SUSP_CTRL register to be 1. The interrupt can be enabled/disabled by setting the bit USB_PHY_CLK_VALID_INT_ENB in USB_SUSP_CTRL to 1/0.

To clear the interrupts USB_WAKEUP_INT_STS and USB_PHY_CLK_VALID_INT_STS, software can write a 1 to corresponding bits.

Generally, whenever PHY is waked up from suspend, first wakeup event is generated (USB_WAKEUP_INT_STS = 1) followed by PHY clock valid interrupt (USB_PHY_CLK_VALID_INT_STS = 1) when PHY clock starts up.

Note: The PLLU_ENABLE bit in PLLU_BASE register should be always set to ENABLE. All the parameters for PLLU in PLLU_BASE and PLLU_MISC register should be set according to the oscillator frequency used. Not doing that will put the USB PHY PLL in a free-running mode, and can result in undesirable side-effects.

When the USB PHY is suspended, the USB3 clock should not be stopped. It can be reduced to 32 KHz, however. If the system clock is stopped, then it may not be possible to bring up the system on a wakeup event described below.

When USB3 is in suspend, AHB clocks to USB3 controller is turned off, and hence there should be no register access to USB2_CONTROLLER_2_* registers. All registers marked as ARUSB3_IF_* are accessible, and can be used to resume the UTMIP3 PHY clocks.

There are two cases to consider for controlling the PHY clocks.

Device Mode

The PHY clock can be turned off in two cases:

- When USB cable isn't connected
- When the USB Host puts the USB bus in suspend mode

When USB cable isn't connected, the VBUS signal is 0. In this case, the software needs to turn on the PHY clock on cable insertion. Follow the procedure described above in the section on Detection of cable insertion.

When USB cable is connected, the PHY clock can only be turned off when the bit SLI in USB2D_USBSTS register is set to 1 when the USB Host puts the USB bus in suspend mode. This bit can be used to generate an interrupt if the bit SLE in USB2D_USBINTR register if set to 1.

In this case, software needs to set two bits in the USB_SUSP_CTRL register to turn on the PHY clock on a USB resume or a USB reset event from USB Host: WAKE_ON_DISCON_EN_DEV: Wake on Disconnect enable during device mode, and USB_WAKE_ON_RESUME_EN - Wake on Resume enable.

If USB_WAKE_ON_RESUME_EN bit in USB_SUSP_CTRL register is set to 1, the PHY clock can be turned on automatically on receiving USB resume event from USB Host. If WAKE_ON_DISCON_EN_DEV bit in USB_SUSP_CTRL register is set to 1, the PHY clock can be turned on automatically on receiving USB reset event from USB Host or if the USB cable is disconnected. These bits should only be turned on after the PHY clock is turned off. The method described above can be used to interrupt the processor when the PHY clock is turned on. Also, turn off these bits when coming out of suspend.

Host Mode

When a device is not connected, software can set the WKCN bit in USB2D_PORTSC1 register. Make sure that VBUS is also turned on. After this, it can stop the PHY clock by using the method described above. The PHY clock will resume automatically

when a device is connected. The method described above can be used to interrupt the processor when the PHY clock is turned on.

With a device connected, software needs to put the USB system under suspend by setting the bit SUSP in USB2D_PORTSC1 register. It needs to wait until this bit is set to 1 by the controller to make sure that the USB system actually went into suspend. It can enable the bits USB_WAKE_ON_RESUME_EN in USB_SUSP_CTRL register and WK_DS in USB2D_PORTSC1 register. Then it can stop the PHY clock as described above. The PHY clock will be turned on automatically if the USB device disconnects or it does a remote wakeup signaling. There will be an interrupt associated with this as described above.

If the software wants to wakeup USB system, then it needs to turn on the PHY clock as described above.

26.3.4.7 Initialization and Shutdown Procedure for USB

1. Bring up USB3 clocks by writing 1 to CLK_ENB_USB3 in ARCLK_RST_CLK_OUT_ENB_H register.
2. Assert and de-assert SWR_USB3_RST in RST_DEVICES_H register to bring the USB block out of reset.
3. Set UTMIP_RESET bit in USB_SUSP_CTRL register to 1 to keep UTMIP3 PHY in reset.
4. Program UTMIP and PLLU parameters.
   - Set register UTMIP_MISC_CFG0 (0x7000:0a24) UTMIP_SUSPEND_EXIT_ON_EDGE (bit 22) to 0.
   - Set the PLLU_ENABLE to 1, and never ever de-assert it.
5. Set UTMIP_RESET bit in USB_SUSP_CTRL register to 0 to bring UTMIP3 out of reset.
   - Wait until PHY_CLKVALID is set to 1.
6. Set USB controller and PHY to suspend by writing 1 to PHCD in USB2D_PORTSC1 register. This will reduce the power consumption on USB3 controller and UTMIP3 PHY.
   - Wait until PHY_CLKVALID is set to 0.
7. Set all PD bits required to bring USB pads to low power mode.
   - UTMIP_OTGPD in UTMIP_BIAS_CFG0 is needed for VBUS detection, and so it should be set to 0.
   - If using VBUS_WAKEUP for VBUS detection, then UTMIP_BIASPD in UTMIP_BIAS_CFG0 can be set to 1 to save power. But if using A_SESS_VLD or any other VBUS sensor for VBUS detection, then this bit should be set to 0.
   - ID_PU in USB_PHY_VBUS_WAKEUP_ID should be set to 1 to enable ID detection.
   - UTMIP_IDPD_SEL in UTMIP_BIAS_CFG0 should be set to 0.
8. To interrupt the processor on VBUS or ID detection, software can set the following:
   - Enable appropriate VBUS interrupt enable, for example, I using A_SESS_VLD sensor for VBUS detection, set A_SESS_VLD_INT_EN in USB_PHY_VBUS_SENSORS register to 1.
   - Enable ID detection interrupt by setting ID_INT_EN in USB_PHY_VBUS_WAKEUP_ID register.
9. When (VBUS=1) or (ID=0) is detected, do the following:
   - Bring USB3 controller and PHY out of suspend by pulsing SUSP_CLR in USB_SUSP_CTRL register by first writing 1 and then writing 0.
     - Wait until PHY_CLKVALID is set to 1.
   - Clear the PD bits that are required for the normal operation.
10. From this on, driver can run the normal operations.
    - UTMIP should only be suspended by writing 1 to PHCD in USB2D_PORTSC1 register when USB cable is connected and only after USB bus is in suspend, for both device and host modes.
11. If a cable disconnect is detected (VBUS = 0) or (ID = 1), then go back to step 4.

26.3.5 BOOTROM Initialization Sequence for USB Recovery

1. Program PLL_U.
   - Setting the PLLU_BASE register fields, PLLU_DIVM, PLLU_DIVN, PLLU_VCO_FREQ, PLLU_BYPASS and PLLU_ENABLE fields as described in the document.
2. Configuring USB1
   - Bring up USB1 clocks by writing 1 to CLK_ENB_USBD in CLK_OUT_ENB_L register.
   - Assert and de-assert master USBD reset in CAR (SWR_USBD_RST in RST_DEVICES_L register) to bring USB1 out of reset.
   - Set USB1_NO_LEGACY_MODE bit in USB1_LEGACY_CTRL register to 1 to use new UTMIP registers.
   - Assert UTMIP_RESET in USB1_IF_USB_SUSP_CTRL register to keep UTMIP1 in reset.
   - Stop crystal clock by setting UTMIP_PHY_XTAL_CLOCKEN in UTMIP_MISC_CFG1 register to 0. This only stops the crystal clocks in the UTMIP units.
     - Default value of the USB1_UTMIP_PHY_XTAL_CLOCKEN is 1, now changing to 0.
     - To Use the A Session Valid for cable detection logic, set USB1_VBUS_SENSE_CTL field in USB1_LEGACY_CTRL register to A_SESS_VLD (2'b11).
   - Programming automatic PLL start times.
     - Setting USB1_IF_UTMIP_PLLU_ENABLE_DLY_COUNT, UTMIP_PLLU_STABLE_COUNT, UTMIP_PLL_ACTIVE_DLY_COUNT and UTMIP_XTAL_FREQ_COUNT as per the values given in the document.
   - Programming the tracking duration.
     - Setting USB1_IF_UTMIP_BIAS_CFG1.UTMIP_BIAS_PDTRK_COUNT as per the values given in the document.
   - Program the debouncer length times.
     - Setting UTMIP_DEBOUNCE_CFG0.UTMIP_BIAS_DEBOUNCE_A field.
   - Program various static parameters of the USB UTMIP1.
     - Set UTMIP_TX_CFG0.UTMIP_FS_PREAMBLE_J to 0x1.
     - Set UTMIP_BAT_CHRG_CFG0.UTMIP_PD_CHRG to 1.
     - Set UTMIP_XCVR_CFG0.UTMIP_XCVR_LSBIAS_SEL to 0.
     - Set 3rd bit of UTMIP_SPARE_CFG0 to 1. i.e UTMIP_SPARE_CFG0 [3] to 1.
     - Set UTMIP_HSRX_CFG0. UTMIP_IDLE_WAIT as per the values given in doc.
     - Set UTMIP_HSRX_CFG0.UTMIP_ELASTIC_LIMIT to 16.
     - Set UTMIP_HSRX_CFG1.UTMIP_HS_SYNC_START_DLY to 9
   - Resuscitate the crystal clock by setting UTMIP_PHY_XTAL_CLOCKEN in UTMIP_MISC_CFG1 register to 1 for USB.
3. CONFIGURE USB3
   - Bring up USB3 clocks by writing 1 to CLK_ENB_USB3 in CLK_OUT_ENB_H register.
   - Assert and de-assert master USB3 reset in CAR (SWR_USB3_RST in RST_DEVICES_H register) to bring USB3 out of reset.
   - Assert UTMIP_RESET in USB3_IF_USB_SUSP_CTRL register to keep UTMIP3 in reset.
   - Change USB3 to use UTMIP PHY by setting UTMIP_CLK_ENB in USB3_IF_USB_SUSP_CTRL register to 1.
   - Stop crystal clock by setting UTMIP_PHY_XTAL_CLOCKEN in UTMIP_MISC_CFG1 register to 0. This only stops the crystal clocks in the UTMIP units.
     - Default value of the USB3_UTMIP_PHY_XTAL_CLOCKEN is 1, now changing to 0.

- Programming automatic PLL start times.
  - Setting USB3_IF_UTMIP_PLLU_ENABLE_DLY_COUNT, UTMIP_PLLU_STABLE_COUNT, UTMIP_PLL_ACTIVE_DLY_COUNT and UTMIP_XTAL_FREQ_COUNT as per the values given in the document.
- Programming the tracking duration.
  - Set when the USB Host puts the USB bus in suspend mode ing USB3_IF_UTMIP_BIAS_CFG1.UTMIP_BIAS_PDTRK_COUNT as per the values given in the document.
- Program the debouncer length times.
  - Setting UTMIP_DEBOUNCE_CFG0.UTMIP_BIAS_DEBOUNCE_A field.
- Program various static parameters of the USB3 UTMIP.
  - Set UTMIP_TX_CFG0.UTMIP_FS_PREAMBLE_J to 0x1.
  - Set UTMIP_BAT_CHRG_CFG0.UTMIP_PD_CHRG to 1.
  - Set UTMIP_XCVR_CFG0.UTMIP_XCVR_LSBIAS_SEL to 0.
  - Set 3rd bit of UTMIP_SPARE_CFG0 to 1. i.e UTMIP_SPARE_CFG0 [3] to 1.
  - Set UTMIP_HSRX_CFG0. UTMIP_IDLE_WAIT as per the values given in doc.
  - Set UTMIP_HSRX_CFG0.UTMIP_ELASTIC_LIMIT to 16.
  - Set UTMIP_HSRX_CFG1.UTMIP_HS_SYNC_START_DLY to 9
- Resuscitate the crystal clock by setting UTMIP_PHY_XTAL_CLOCKEN in UTMIP_MISC_CFG1 register to 1 for USB3.
- Bring up the UTMIP clock for USB3
  - De-assert UTMIP_RESET in USB3_IF_USB_SUSP_CTRL register to bring UTMIP3 out of reset.
  - Wait until USB3_IF_USB_SUSP_CTRL.PHY_CLK_VALID is set to 1.
- Program USB3 controller to use UTMIP interface.
  - Disable ICUSB interface (disable by default) by writing IC_ENB1 field in USB2_CONTROLLER_2_USB2D_ICUSB_CTRL_0 register.
  - Change USB3 controller to use UTMIP PHY by setting STS field to 0 and PTS field to 1 in USB2_CONTROLLER_2_USB2D_PORTSC1 register.
- Hold the reset for UTMIP3.
  - Set USB3 PHY (UTMIP3) to reset by writing 1 to UTMIP_RESET in USB3_IF_USB_SUSP_CTRL register.
  - Wait until USB3_IF_USB_SUSP_CTRL.PHY_CLK_VALID is set to 0.

4. Waiting for cable connect on USB UTMIP1 port. When cable is connected on USB UTMIP1 port,
5. Bring UTMIP1 out of reset by writing 0 to UTMIP_RESET bit of USB1_IF_SUSP_CTRL register.
6. Wait until USB1_IF_USB_SUSP_CTRL.PHY_CLK_VALID is set to 1.
7. Then perform USB controller initialization.
   - Do Bus reset.
   - Wait until bus come out of reset.
   - Set the controller in device mode.
   - USB operations.

26.3.6 Performance Settings for USB Controllers

To meet USB's strict bandwidth/latency requirements, some AHB programming needs to be done. The following gives a guideline on the programming requirements to achieve maximum performance from USB:

- The burst size for USB controller should be programmed to 8 in the register USB2D_BURSTSIZE. Both TXPBURST and RXPBURST fields should be programmed to the same value of 8.
- The field ENB_FAST_REARBITRATE for AHB_MEM gizmo should be set to 1 in the register AHB_GIZMO_AHB_MEM.
- The field IMMEDIATE for USB gizmos should be set to 1 in the register AHB_GIZMO_USB, AHB_GIZMO_USB2 or AHB_GIZMO_USB3 depending on the controller in use.
- USB controllers should be set as high-priority masters on AHB by setting the bits corresponding to each USB controller to 1 in AHB_PRIORITY_SELECT field in the register AHB_ARBITRATION_PRIORITY_CTRL and setting the priority weight to 7 by setting the field AHB_PRIORITY_WEIGHT in the same register. USB master numbers are 6 for USB1, 18 for USB2 and 17 for USB3. The priority weight could be relaxed depending on requirements from other AHB masters in the system as required for different use cases.
- Prefetch engine needs to be setup correctly to enable prefetching of transmit data packets for USB masters. Each USB master needs one channel on the prefetch engine. There are 4 channels on the prefetch engine. If all 3 USB masters enable one channel at the same time, it would leave one more channel for another AHB master. Each prefetch channel is controlled by the register AHB_AHB_MEM_PREFETCH_CFG[NO] where NO=1,2,3,4. To enable prefetch for a USB controller on a channel, program AHB_MST_ID_USB, AHB_MST_ID_USB2 or AHB_MST_ID_USB3 in the field AHB_MST_ID for the register AHB_AHB_MEM_PREFETCH_CFG[NO]. The field ADDR_BNDRY should be set to log2 (buffer size) according to the buffer size required for the corresponding USB master. The field SPEC_THROTTLE should be set to 0, and the field INACTIVITY_TIMEOUT should be set to 0x800.

All these programming need to be done before setting RS bit in USB2D_USBCMD to RUN (1) for the corresponding USB controller.

26.4 UTMIP Programming Guidelines

26.4.1 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.

26.4.2 PLL_U Programming

The USB ports use a cascaded PLL scheme to guarantee a high clock quality. First there is a PLL_U, which is the source clock for both the USB PLLs that are dedicated to ports. PLL_U (of type CLKPLL960_USB) produces reference clocks for all forms of USB (ULPI (null mode), IC USB, UTMIP, UHSIC).

Table 83 describes the parameter settings that are dependent on the crystal clock reference frequency. The parameters are specified in decimal notation.

Table 83 480MHz PLL_U Output Frequency

Fi	13.0MHz	19.2MHz	12.0MHz	26.0MHz
N (PLLU_DIVN)	960	200	960	960
M (PLLU_DIVM)	13	4	12	26

These parameters can be found in the PLLU_BASE register. The PLLU_OVERRIDE bit should be set to 0. Please also note that PLLU_VCO_FREQ parameter must be set to 0. DO NOT change the parameters of PLL_U while the unit is running. The same applies to the USB_PHY_PLL.

26.4.3 PLL_U and USB_PHY_PLL automatic startup times

The PLL_U and USB_PHY_PLL automatic startup times are used in reset and suspend modes to kick start the PLLs. Software should not manually force the PLL up and down as it cannot do so rapidly enough to meet the protocol.

Table 84 PLL Automated Start Times

CRYSTAL FREQ.	13.0 MHz	19.2 MHz	12.0 MHz	26.0 MHz
PLLU_ENALBLE_DELAY_COUNT	2	3	2	4
PLLU_STABLE_COUNT	51	75	47	102
PLL_ACTIVE_DLY_COUNT	5	6	4	9
XTAL_FREQ_COUNT	127	187	118	254

The PLLU_ENABLE_DLY_COUNT should be for at least 1 microsecond, the PLLU_STABLE_COUNT should be for at least 1ms, the PLL_ACTIVE_DLY_COUNT for at least 5 microseconds, and the XTAL_FREQ_COUNT for about 2.5ms.

26.4.4 Powering down a USB port outside of Deep Power Down (DPD)

When a port is known to be disabled, there needs to be a mechanism to stop its power consumption for situations outside of DPD. This section addresses disabling a port while powered up.

Powering down of USB at times other than deep power down should be done by setting all specialized PLL and pad power down pins instead of using the global E_DPD pins of the USB analog components. It has been determined that it is safer to power down the cells through the traditional power down pins when the 3.3V and 1.2V supplies might still be on. This is achieved by setting the following pad controls from UTMIP register space. Before setting the power down bits; save previous registers values so that they may be restored. This type of power down should not be done in the Boot ROM.

Table 85 UTMIP Pad Controls

Pad Control	Analog Cell	Register Bit Settings
PD	Transceiver	UTMIP_FORCE_PD_POWER_DOWN=1
PD2	Transceiver	UTMIP_FORCE_PD2_POWER_DOWN=1
PD_ZI	Transceiver	UTMIP_FORCE_PDZI_POWER_DOWN=1
PD_DR	Transceiver	UTMIP_FORCE_PDDR_POWER_DOWN=1
PD_CHRP	Transceiver	UTMIP_FORCE_PDCHRP_POWER_DOWN=1
PD_DISC	Transceiver	UTMIP_FORCE_PDDISC_POWER_DOWN=1
PD_CHG	Transceiver	UTMIP_PD_CHG=1
PD	Bias	UTMIP_BIASPD=1
PD_TRK	Bias	UTMIP_FORCE_PDTRK_POWER_DOWN=1
OTG_PD[X]	Bias	UTMIP_OTGPD=1
ID_PD[X]	Bias	UTMIP_IDPD_SEL=1, UTMIP_IDPD_VAL=1
VBUS_WAKEUP_PD[X]	Bias	UTMIP_VBUS_WAKEUP_POWER_DOWN=1
ENABLE	PHY PLL	UTMIP_FORCE_PLL_ENABLE_POWERDOWN=1
ACTIVE	PHY PLL	UTMIP_FORCE_PLL_ACTIVE_POWERDOWN=1

Pad Control	Analog Cell	Register Bit Settings
ENABLE	PLL U	UTMIP_FORCE_PLLU_POWERDOWN=1
UTMIP	PHY	UTMIP_PHY_XTAL_CLOCKEN=0

The set of bits exists for both ports. Three of the power downs are explicitly labeled with a port index because they belong to the bias cell and happen to have a per port control. Otherwise the same register bits exist in both controllers.

26.4.5 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.

26.4.6 Deep Power Down Behavior

Tegra 2 Series devices deep power down behavior for USB wake-up events is controlled in the PMC via registers USB_DEBOUNCE_DLY, USB_AO and the WAKE_MASK (USB_EVENT field) of the PMC unit. These control the wake-up events, their debounce duration and their debounce length while powered down. Each USB port has a possible event on ID or VBUS detection. These can be configured at startup time. These registers take over from the UTMIP registers when the chip is powered down. The programming depends on the context of the chip. For example, if both USB ports are dedicated host ports then the VBUS wake-up event should be kept powered down. Conversely a dedicated device mode port may not want to wake-up on the detection of an ID connector and can chose to power down ID.

26.5 USB Registers

26.5.1 USB1 Controller Registers

26.5.1.1 USB2_CONTROLLER_USB2D_ID_0

USB2D Identification Register

Offset: 000h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	REVISION: Revision number of the USB controller. This is set to 0x33.
15:8	X	NID: Ones complement version of ID. This field is set to 0xFA.
7:0	X	ID: Configuration number. This field is set to 0x05

26.5.1.2 USB2_CONTROLLER_USB2D_HW_HOST_0

USB2D Hardware Host Register

Offset: 008h | Read/Write: RO | Reset: 0bxxxx

Bit	Reset	Description
3:1	X	NPORT: VUSB_HS_NUM_PORT-1: This host controller has only 1 port. So this field will always be 0.
0	X	HC: VUSB_HS_HOST: Indicates support for host mode. Set to 1.

26.5.1.3 USB2_CONTROLLER_USB2D_HW_DEVICE_0

USB2D Hardware Device Register

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

Bit	Reset	Description
5:1	X	DEVEP: VUSB_HS_DV_EP: No. of endpoints supported by this device controller. Set to 16. This includes control endpoint 0.
0	X	DC: Device capable: Set to 1 indicating support for device mode.

26.5.1.4 USB2_CONTROLLER_USB2D_HW_TXBUF_0

USB2D Hardware TX Buffer Register

Offset: 010h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	TXCHANADD: VUSB_HS_TX_CHAN_ADD: Total no. of address bits for the transmit buffer of each transmit endpoint. Set to 7. Each transmit buffer is 128 words deep.
15:8	X	TXADD: VUSB_HS_TX_ADD: Total no. of address bits for the transmit buffer. Set to 11. The total depth of the transmit buffer is 2048 words.
7:0	X	TCBURST: VUSB_HS_TX_BURST: Maximum burst size supported by the transmit endpoints for data transfers. Set to 8.

26.5.1.5 USB2_CONTROLLER_USB2D_HW_RXBUF_0

USB2D RX Buffer HW Parameters Register

Offset: 014h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:8	X	RXADD: VUSB_HS_RX_ADD: Total no. of address bits for the receive buffer. Set to 7. The total depth of the receive buffer is 128 words
7:0	X	RXBURST: VUSB_HS_RX_BURST: Maximum burst size supported by the receive endpoints for data transfers. Set to 8.

26.5.1.6 USB2_CONTROLLER_USB2D_CAPLENGTH_0

USB2D Capability Register Length Register

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

Bit	Reset	Description
7:0	X	CAPLENGTH: Indicates which offset to add to the register base address at the beginning of the Operational Register. Set to 0x40.

26.5.1.7 USB2_CONTROLLER_USB2D_HCIVERSON_0

USB2D Host Interface Version Number Register

Offset: 102h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	HCIVERSION: Contains a BCD encoding of the EHCI revision number supported by this host controller. The most significant byte of this register represents a major revision and the least significant byte is the minor revision. This host controller supports EHCI revision 1.00.

26.5.1.8 USB2_CONTROLLER_USB2D_HCSPARAMS_0

USB2D Host Control Structural Parameters Register

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

Bit	Reset	Description
27:24	X	N_TT: Number of Transaction Translators: indicates the number of embedded transaction translators associated with the USB2.0 host controller. This field is always set to 1 indicating only 1 embedded TT is implemented in this implementation. This is a non-EHCI field to support embedded TT.
23:20	X	N_PTT: Number of Ports per Transaction Translator: indicates the number of ports assigned to each transaction translator within the USB2.0 host controller. Field always equals N_PORTS. This is a non-EHCI field to support embedded TT.
15:12	X	N_CC: Number of Companion Controller: indicates the number of companion controllers. This field is set to 0.
11:8	X	N_PCC: Number of Ports per Companion Controller: indicates the number of ports supported per internal companion controller. This field is set to 0.
4	X	PPC: Port Power Control: indicates whether the host controller implementation includes port power control. 1 = Ports have port power switches 0= Ports do not have port power switches. This field affects the functionality of the port Power field in each port status and control register. This field is set to 1.
3:0	X	N_PORTS: Number of downstream ports. This field specifies the number of physical downstream ports implemented on this host controller. This field is fixed to 1, since this host controller only supports 1 port.

26.5.1.9 USB2_CONTROLLER_USB2D_HCCPARAMS_0

USB2D Host Control Capability Parameters Register

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

Bit	Reset	Description
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".

26.5.1.10 USB2_CONTROLLER_USB2D_DCIVERSION_0

USB2D Device Interface Version Number Register

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

Bit	Reset	Description
15:0	X	DCIVERSION: The device controller interface conforms to the two-byte BCD encoding of the interface version number contained in this register.

26.5.1.11 USB2_CONTROLLER_USB2D_DCCPARAMS_0

USB2D Device Control Capabilities Register

Offset: 124h | Read/Write: RO | Reset: 0bxxxxxxxxx

Bit	Reset	Description
8	X	HC: Host Capable: 1 = This controller is capable of operating as an EHCI compatible USB 2 0 host controller operating as an EHCI compatible USB 2.0 host controller. This field is set to 1.
7	X	DC: Device Capable: 1 = Controller is capable of operating as USB 2.0 device. This field is set to 1.
4:0	X	DEN: Device Endpoint Number: Number of endpoints built into the device controller. This is set to 16.

26.5.1.12 USB2_CONTROLLER_USB2D_USBCMD_0

USB2D USB Command Register

Offset: 140h | Read/Write: R/W | Reset: 0b00001000000x1x11x0000000

Bit	R/W	Reset	Description
23:16	RW	0x8	ITC: Interrupt Threshold Control .Read/Write. Default 08h. The system software uses this field to

Bit	R/W	Reset	Description
			set the maximum rate at which the host/device controller will issue interrupts. ITC contains the maximum interrupt interval measured in micro-frames. Valid values are shown below. Value Maximum Interrupt Interval 00h Immediate (no threshold) 01h 1 micro-frame 02h 2 micro-frames 04h 4 micro-frames 08h 8 micro-frames 10h 16 micro-frames 20h 32 micro-frames 40h 64 micro-frames 0 = IMMEDIATE 2 = ONE_MF 4 = TWO_MF 8 = EIGHT_MF 16 = SIXTEEN_MF 32 = THIRTY_TWO_MF 64 = SIXTY_FOUR_MF
15	RW	0x0	FS2: Bit 2 of Frame List Size.
14	RW	0x0	ATDTW: Add DTD Tripwire. This bit is used as a semaphore when a dTD is added to an active (primed) endpoint. This bit is set and cleared by software and will be cleared by hardware when a hazard exists such that adding a dTD to a primed endpoint may go unnoticed. 0 = CLEAR 1 = SET
13	RW	0x0	SUTW: Setup Tripwire. This bit is used as a semaphore when the 8 bytes of setup data read extracted by the firmware. If the setup lockout mode is off, then there exists a hazard when new setup data arrives and firmware is copying setup data from the QH for a previous setup packet. This bit is set and cleared by software and will be cleared by hardware when a hazard exists. 0 = CLEAR 1 = SET
11	RW	0x1	ASPE: Asynchronous Schedule Park mode Enable. Software uses this bit to enable or disable Park mode. When this bit is one, Park mode is enabled. When this bit is a zero, Park mode is disabled. This field is set to "1" in this implementation. 0 = DISABLE 1 = ENABLE
9:8	RW	0x3	ASP1_ASP0: Asynchronous Schedule Park Mode Count (OPTIONAL) Read/Write. If the Asynchronous Park Capability bit in the HCCPARAMS register is a one, then this field defaults to 3h and is R/W. Otherwise it defaults to zero and is RO. It contains a count of the number of successive transactions the host controller is allowed to execute from a high-speed queue head on the Asynchronous schedule before continuing traversal of the Asynchronous schedule. Valid values are 1h to 3h. Software must not write a zero to this bit when Park Mode Enable is a one as this will result in undefined behavior. This field is set to 3h in this implementation and is Read/Write capable.
7	RO	X	LR: Light Host/Device Controller Reset (OPTIONAL) . Read Only. Not Implemented. This field will always be "0".
6	RW	0x0	IAA: Interrupt on Async Advance Doorbell. When the host controller has evicted all appropriate cached schedule states, it sets the Interrupt on Async Advance status bit in the USBSTS register. If the Interrupt on Sync Advance Enable bit in the USBINTR register is one, then the host controller will assert an interrupt at the next interrupt threshold. The host controller sets this bit to zero after it has set the Interrupt on Sync Advance status bit in the USBSTS register to one. Software should not write a one to this bit when the asynchronous schedule is inactive. Doing so will yield undefined results. This bit is only used in host mode. Writing a one to this bit when device mode is selected will have undefined results. 0 = CLEAR 1 = SET
5	RW	0x0	ASE: Asynchronous Schedule Enable. This bit controls whether the host controller skips processing the Asynchronous Schedule. 0 = Do not process the Asynchronous Schedule. 1 = Use the ASYNCLISTADDR register to access the Asynchronous Schedule. Only the host controller uses this bit. 0 = DISABLE 1 = ENABLE
4	RW	0x0	PSE: Periodic Schedule Enable.This bit controls whether the host controller skips processing the 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

Bit	R/W	Reset	Description
			1 = ENABLE
3:2	RW	0x0	FS1_FS0: Frame List Size . (Read/Write). 000 = Default This field is Read/Write only if Programmable Frame List Flag in the HCCPARAMS registers is set to one. Hence this field is Read/Write for this implementation. This field specifies the size of the frame list that controls which bits in the Frame Index Register should be used for the Frame List Current index. Note that this field is made up from USBCMD bits 15, 3 and 2. 000 = 1024 elements (4096 bytes) Default value 001 = 512 elements (2048 bytes) 010 = 256 elements (1024 bytes) 011 = 128 elements (512 bytes) 100 = 64 elements (256 bytes) 101 = 32 elements (128 bytes) 110 = 16 elements (64 bytes) 111 = 8 elements (32 bytes) Only the host controller uses this field.
1	RW	0x0	RST: Controller Reset. Software uses this bit to reset the controller. This bit is set to zero by the Host/Device Controller when the reset process is complete. Software cannot terminate the reset process early by writing a zero to this register. Host Controller: When software writes a one to this bit, the Host Controller resets its internal pipelines, timers, counters, state machines etc. to their initial value. Any transaction currently in progress on USB is immediately terminated. A USB reset is not driven on downstream ports. Software should not set this bit to a one when the HCHalted bit in the USBSTS register is a zero. Attempting to reset an actively running host controller results in undefined behavior. Device Controller: When software writes a one to this bit, the Device Controller resets its internal pipelines, timers, counters, state machines etc. to their initial value. Any transaction currently in progress on USB is immediately terminated. Writing a one to this bit in device mode is not recommended. 0 = CLEAR 1 = SET
0	RW	0x0	RS: Run/Stop: Host Controller: When set to a 1, the Host Controller proceeds with the execution of the schedule. The Host Controller continues execution as long as this bit is set to a one. When this bit is set to 0, the Host Controller completes the current transaction on the USB and then halts. The HCHalted bit in the status register indicates when the Host Controller has finished the transaction and has entered the stopped state. Software should not write a one to this field unless the host controller is in the Halted state (i.e. HCHalted in the USBSTS register is a one). Device Controller: Writing a one to this bit will cause the device controller to enable a pull-up on D+ and initiate an attach event. This control bit is not directly connected to the pull-up enable, as the pull-up will become disabled upon transitioning into high-speed mode. Software should use this bit to prevent an attach event before the device controller has been properly initialized. Writing a 0 to this will cause a detach event. 0 = STOP 1 = RUN

26.5.1.13 USB2_CONTROLLER_USB2D_USBSTS_0

USB2D USB Status Register

Offset: 144h | Read/Write: R/W | Reset: 0b00x1xxx0000x0000

Bit	R/W	Reset	Description
15	RW	0x0	AS: Asynchronous Schedule Status. This bit reports the current real status of the Asynchronous Schedule. When set to zero the asynchronous schedule status is disabled and if set to one the status is enabled. The Host Controller is not required to immediately disable or enable the Asynchronous Schedule when software transitions the Asynchronous Schedule Enable bit in the USBCMD register. If AS = ASE: 1= Enable Asynchronous Schedule 0= Disable Asynchronous Schedule Only used by the host controller. 0 = DISABLE 1 = ENABLE
14	RW	0x0	PS: Periodic Schedule Status. This bit reports the current real status of the Periodic Schedule. When set to zero the periodic schedule is disabled, and if set to one the status is enabled. The Host Controller is not required to immediately disable or enable the Periodic Schedule when software transitions the Periodic Schedule Enable bit in the USBCMD register. If PS = PSE then: 1 = Periodic Schedule is enabled or 0 = Periodic Schedule is disabled Only used by the host controller. 0 = DISABLE 1 = ENABLE
13	RO	X	RCL: Reclamation. This is a read-only status bit used to detect an empty asynchronous schedule. Only used by the host controller. 0 = DISABLE

Bit	R/W	Reset	Description
			1 = ENABLE
12	RW	0x1	HCH: HCHalted. 1 = Default. This bit is a zero whenever the Run/Stop bit is a one. The Host Controller sets this bit to one after it has stopped executing because of the Run/Stop bit being set to 0, either by software or by the Host Controller hardware (e.g. internal error). Only used by the host controller. 0 = UNHALTED 1 = HALTED
8	RW	0x0	SLI: DCSuspend. When a device controller enters a suspend state from an active state, this bit will be set to a 1. The device controller clears the bit upon exiting from a suspend state. Only used by the device controller. 0 = NOTSUSPEND 1 = SUSPENDED
7	RW	0x0	SRI: SOF Received. When the device controller detects a Start Of (micro) Frame, this bit will be set to a one. When a SOF is extremely late, the device controller will automatically set this bit to indicate that an SOF was expected. Therefore, this bit will be set roughly every 1ms in device FS mode and every 125us in HS mode and will be synchronized to the actual SOF that is received. Since device controller is initialized to FS before connect, this bit Will be set at an interval of 1ms during the prelude to the connect and chirp. In host mode, this bit will be set every 125us and can be used by host controller driver as a time base. Software writes a 1 to this bit to clear it. This is a non-EHCI status bit. 0 = SOF_NOT_RCVD 1 = SOF_RCVD
6	RW	0x0	URI: USB Reset Received. When the device controller detects a USB Reset and enters the default state, this bit is set to a 1. Software can write a 1 to this bit to clear the USB Reset Received status bit. Only used by the device controller. 0 = NO_USB_RESET 1 = USB_RESET
5	RW	0x0	AAI: Interrupt and Asynchronous Advance. System software can force the host controller to issue an interrupt the next time the host controller advances the asynchronous schedule by writing a one to the Interrupt on Async Advance Doorbell bit in the USBCMD register. This status bit indicates the assertion of that interrupt source. Only used by the host controller 0 = NOT_ADVANCED 1 = ADVANCED
4	RO	X	SEI: System Error. This bit is not used in this implementation and will always be set to "0". 0 = NO_ERROR 1 = ERROR
3	RW	0x0	FRI: Frame List Rollover. The Host Controller sets this bit to a 1 when the Frame List Index rolls over from its maximum value to 0. The exact value at which the rollover occurs depends on the frame list size. For example. If the frame list size (as programmed in the Frame List Size field of the USBCMD register) is 1024, the Frame Index Register rolls over every time FRINDEX [1 3] toggles. Similarly, if the size is 512, the Host Controller sets this bit to a 1 every time FHINDEX [12] toggles. Only used by the host controller. 0 = NO_ROLLOVER 1 = ROLLOVER
2	RW	0x0	PCI: Port Change Detect. The Host Controller sets this bit to a 1 when on any port a Connect Status occurs, a Port Enable/Disable Change occurs, or the Force Port Resume bit is set as the result of a J-K transition on the suspended port. The Device Controller sets this bit to a one when the port controller enters the full or high-speed operational state. When the port controller exits the full or high-speed operational states due to Reset or Suspend events, the notification mechanisms are the USB Reset Received bit and the DCSuspend bits respectively. This bit is not EHCI compatible. 0 = NO_PORT_CHANGE 1 = PORT_CHANGE
1	RW	0x0	UEI: USB Error Interrupt. This bit gets set by the Host/Device controller when completion of a USB transaction results in an error condition. This bit is set along with the USBINT bit, if the TD on which the error interrupt occurred also ad its interrupt on complete (IOC) bit set. 0 = NO_ERROR 1 = ERROR
0	RW	0x0	UI: USB Interrupt. This bit is set by the Host/Device Controller when the cause of an interrupt is a

Bit	R/W	Reset	Description
			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

26.5.1.14 USB2_CONTROLLER_USB2D_USBINTR_0

USB2D USB Interrupt Enable Register

Offset: 148h | Read/Write: R/W | Reset: 0b000000000

Bit	Reset	Description
8	0x0	SLE: Sleep Enable. 1 = Device controller issues an interrupt if DCSuspend bit in USBSTS register transitions. The interrupt is acknowledged by SW by writing a 1 to the DCSuspend bit. Only used by the device controller. 0 = DISABLE 1 = ENABLE
7	0x0	SRE: SOF Received Enable. 1 = Device controller issues an interrupt if SOF Received bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the SOF Received bit. 0 = DISABLE 1 = ENABLE
6	0x0	URE: USB Reset Enable.1 = Device controller issues an interrupt if USB Reset Received bit in USBSTS register = 1 The interrupt is acknowledged by software clearing the USB Reset Received bit. Only used by the device controller. 0 = DISABLE 1 = ENABLE
5	0x0	AAE: Interrupt on Asynchronous Advance Enable. 1 = the host controller issues an interrupt at the next interrupt threshold if Interrupt on Async Advance bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the Interrupt on Async Advance bit. Only used by the host controller. 0 = DISABLE 1 = ENABLE
4	0x0	SEE: System Error Enable. 1 = Host/device controller issues an interrupt if the System Error bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the System Error bit. 0 = DISABLE 1 = ENABLE
3	0x0	FRE: Frame List Rollover Enable. 1 = Host controller issues an interrupt if Frame List Rollover bit in the USBSTS register = 1. The interrupt is acknowledged by software clearing the Frame List Rollover bit. Only used by the host controller. 0 = DISABLE 1 = ENABLE
2	0x0	PCE: Port Change Detect Enable. 1 = Host/device controller issues an interrupt if Port Change Detect bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the Port Change Detect bit. 0 = DISABLE 1 = ENABLE
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

26.5.1.15 USB2_CONTROLLER_USB2D_FRINDEX_0

USB2D USB Frame Index Register

Offset: 14ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxx

Bit	Reset	Description
13:0	X	FRINDEX: Frame Index. The value in this register increments at the end of each time frame (micro-frame). Bits [N: 3] are used for the Frame List current index. Each location of the frame list is accessed 8 times (frames or micro-frames) before moving to the next index. The following illustrates values of N based on the value of the Frame List Size field in the USBCMD register, when used in host mode. USBCMD [Frame List Size] Number Elements N 000b (1024) 12 001b (512) 11 010b (256) 10 011b (128) 9 100b (64) 8 101b (32) 7 110b (16) 6 111b (8) 5 In device mode the value is the current frame number of the last frame transmitted. It is not used as an index. In either mode bits 2:0 indicate the current micro-frame.

26.5.1.16 USB2_CONTROLLER_USB2D_PERIODICLISTBASE_0

USB2D Host Controller Frame List Base Address Register

Offset: 154h | Read/Write: R/W | Reset: 0b00000000000000000000

Bit	Reset	Description
31:25	0x0	USBADR: Device mode. The upper seven bits of this register represent the device address. After any controller reset or a USB reset, the device address is set to the default address (0). The default address will match all incoming addresses. Software shall reprogram the address after receiving a SET_ADDRESS request.
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.

26.5.1.17 USB2_CONTROLLER_USB2D_ASYNCLISTADDR_0

USB2D Next Asynchronous List Address Register

Offset: 158h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:11	0x0	EPBASE: Device mode. This register contains the address of the top of the endpoint list in system memory. These bits correspond to memory address signals [31:11], respectively. This field will reference a list of up to 32 Queue Heads (QH). Only used by the device controller.
31:5	0x0	ASYBASE: Host mode. This 32-bit register contains the address of the next asynchronous queue head to be executed by the host. Link Pointer Low (LPL). These bits correspond to memory address signals [31:5], respectively. This field may only reference a Queue Head (OH). Only used by the host controller.

26.5.1.18 USB2_CONTROLLER_USB2D_ASYNCTTSTS_0

USB2D Asynchronous Buffer Status for Embedded TT Register

Offset: 15ch | Read/Write: R/W | Reset: 0b0x

Bit	R/W	Reset	Description
1	RW	0x0	TTAC: Embedded TT Async Buffers Clear. (Read/Write to set) This field will clear all pending transactions in the embedded TT Async Buffer(s). The clear will take as much time as necessary to clear buffer without interfering with a transaction in progress. TTAC will return to zero after being set by software only after the actual clear occurs.

Bit	R/W	Reset	Description
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.

26.5.1.19 USB2_CONTROLLER_USB2D_BURSTSIZE_0

USB2D Burst Size register

Offset: 160h | Read/Write: R/W | Reset: 0b0000100000001000

Bit	Reset	Description
15:8	0x8	TXPBURST: Programmable TX Burst Length. (Read/Write) This register represents the maximum length of a burst in 32-bit words while moving data from system memory to the USB bus.
7:0	0x8	RXPBURST: Programmable RX Burst Length. (Read/Write) This register represents the maximum length of a burst in 32-bit words while moving data from the USB bus to system memory.

26.5.1.20 USB2_CONTROLLER_USB2D_TXFILLTUNING_0

USB2D Transmit fill tuning register

Offset: 164h | Read/Write: R/W | Reset: 0b000010xxx0000000000000

Bit	Reset	Description
21:16	0x2	TXFIFOTHRES: FIFO Burst Threshold. (Read/Write) This register controls the number of data bursts that are posted to the TX latency FIFO in host mode before the packet begins on to the bus. The minimum value is 2 and this value should be a low as possible to maximize USB performance. A higher value can be used in systems with unpredictable latency and/or insufficient bandwidth where the FIFO may underrun because the data transferred from the latency FIFO to USB occurs before it can be replenished from system memory. This value is ignored if the Stream Disable bit in USBMODE register is set.
12:8	0x0	TXSCHHEALTH: Scheduler Health Counter. (Read/Write To Clear) [Default = 0] This register increments when the host controller fails to fill the TX latency FIFO to the level programmed by TXFIFOTHRES before running out of time to send the packet before the next Start-Of-Frame. This health counter measures the number of times this occurs to provide feedback to selecting a proper TXSCHOH. Writing to this register will clear the counter and this counter will max. at 31.
7:0	0x0	TXSCHOH: Scheduler Overhead. (Read/Write) [Default = 0] This register adds an additional fixed offset to the schedule time estimator described above as Tff. As an approximation, the value chosen for this register should limit the number of back-off events captured in the TXSCHHEALTH to less than 10 per second in a highly utilized bus. Choosing a value that is too high for this register is not desired as it can needlessly reduce USB utilization. The time unit represented in this register is 1.267us when a device is connected in High-Speed Mode. The time unit represented in this register is 6.333us when a device is connected in Low/Full Speed Mode

26.5.1.21 USB2_CONTROLLER_USB2D_PORTSC1_0

USB2D Port Status/Control 1 Register

Offset: 184h | Read/Write: R/W | Reset: 0bxxxxxxx0x0000000xxx1xxx0x0xx010x

Bit	R/W	Reset	Description
31	RO	X	PTS: 0 = UTMI interface. This is the only value supported. This bit is not defined in the EHCI specification. 0 = UTMI 1 = RESERVED
30	RO	X	STS: 0 = Serial transceiver not selected. This is the only value supported. This bit is not defined in the EHCI specification.

Bit	R/W	Reset	Description
			0 = PARALLEL_IF 1 = SERIAL_IF
29	RO	X	PTW: Parallel Transceiver Width. Fixed to 0. This bit is not defined in the EHCI specification. 0 = EIGHT_BIT 1 = RESERVED
27:26	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	PFSC: Port Force Full Speed Connect: Writing this bit to a 1b forces the port to connect at Full Speed only. It disables the chirp sequence that allows the port to identify itself as High Speed. This is useful for testing FS configurations with a HS host, hub or device. This bit is not defined in the EHCI specification. 0 = DONT_FORCE_FULL_SPEED 1 = FORCE_FULL_SPEED
22	RW	0x0	WKOC: Default = 0b. Wake on Over-current Enable: Writing this bit to a one enables the port to be sensitive to over-current conditions as wake-up events. This field is zero if Port Power(PP) is zero. This bit should only be used when operating in Host mode. Writing this bit to 1 while the controller is working in device mode can result in undefined behavior. 0 = DISBLE 1 = ENABLE
21	RW	0x0	WKDS: Wake on Disconnect Enable: Writing this bit to a one enables the port to be sensitive to device disconnects as wake-up events. This field is zero if Port Power(PP) is zero or in device mode. This bit should only be used when operating in Host mode. Writing this bit to 1 while the controller is working in device mode can result in undefined 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 = DISBLE 1 = ENABLE
20	RW	0x0	WKCN: Wake on Connect Enable: Writing this bit to a one enables the port to be sensitive to device connects as wake-up events. This field is zero if Port Power(PP) is zero or in device mode. This bit should only be used when operating in Host mode. Writing this bit to 1 while the controller is working in device mode can result in undefined 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 = DISBLE 1 = ENABLE
19:16	RW	0x0	PTC: Port Test Control: Any other value than zero indicates that the port is operating in test mode. Value Specific Test 0000b Not enabled 0001b J_ STATE 0010b K_STATE 0011b SEQ_NAK 0100b Packet 0101b FORCE_ENABLE 0110b to 1111b Reserved Refer to Chapter 7 of the USB Specification Revision 2.0 for details on each test mode. 0 = NORMAL_OP 1 = TEST_J 2 = TEST_K 3 = TEST_SE0_NAK 4 = TEST_PKT 5 = TEST_FORCE_ENABLE
15:14	RO	X	PIC: Port Indicator Control: This field is not supported in the current implementation. Please use a GPIO if you wish to use Port Indicators.
13	RO	X	PO: Port Owner. Port owner handoff is not implemented in this design, therefore this bit will 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 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

Bit	R/W	Reset	Description
			port is non-functional and will not report attaches, detaches, etc. When an over-current condition is detected on a powered port and PPC is a one, the PP bit in each affected port may be transitioned by the host controller driver from a one to a zero (removing power from the port). 0 = NOT_POWERED 1 = POWERED
11:10	RO	X	LS: Line state. These bits reflect the current logical levels of the D+ (bit 10) and D- (bit 11) signal lines. The encoding of the bits are: 00b = SE0 01b = J-state 10b = K-state 11b = Undefined The value of this field is undefined if Port Power(PP) is zero in host mode. In host mode, the use of line-state by the host controller driver is not necessary (unlike EHCI), because the port controller state machine and the port routing manage the connection of LS and FS. In device mode, the use of line-state by the device controller driver is not necessary. 0 = SE0 1 = J_STATE 2 = K_STATE 3 = UNDEFINED
9	RO	X	HSP: When the bit is one, the host/device connected to the port is in high-speed mode and if set to zero, the host/device connected to the port is not in a high-speed mode. Note: HSP is redundant with PSPD(27:26). This bit is not defined in the EHCI specification. 0 = NOT_HIGH_SPEED 1 = HIGH_SPEED
8	RW	0x0	PR: This field is zero if Port Power(PP) is zero. In Host Mode: Read/Write. 1=Port is in Reset. 0=Port is not in Reset. When software writes a one to this bit the bus-reset sequence as defined in the USB Specification Revision 2.0 is started. This bit will automatically change to zero after the reset sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the reset duration is timed in the driver. In Device Mode: This bit is a read only status bit. Device reset from the USB bus is also indicated in the USBSTS register. 0 = NOT_USB_RESET 1 = USB_RESET
7	RO	X	SUSP: Port suspend. 1=Port in suspend state. 0=Port not in suspend state. In Host Mode: Read/Write. Port Enabled Bit and Suspend bit of this register define the port states as follows: Bits [Port Enabled, Suspend] Port State 0x Disable 10 Enable 11 Suspend When in suspend state, downstream propagation of data is blocked on this port, except for port reset. The blocking occurs at the end of the current transaction if a transaction was in progress when this bit was written to 1. In the suspend state, the port is sensitive to resume detection. Note that the bit status does not change until the port is suspended and that there may be a delay in suspending a port if there is a transaction currently in progress on the USB. The host controller will unconditionally set this bit to zero when software sets the Force Port Resume bit to zero. A write of zero to this bit is ignored by the host controller. If host software sets this bit to a one when the port is not enabled (i.e. Port enabled bit is a zero) the results are undefined. This field is zero if Port Power(PP) is zero in host mode. In Device Mode: Read Only. This bit is a read only status bit. 0 = NOT_SUSPEND 1 = SUSPEND
6	RW	0x0	FPR: Force Port Resume. 1= Resume detected/driven on port. 0=No resume (K state) detected/driven on port. In Host Mode: Software sets this bit to one to drive resume signaling. The Host Controller sets this bit to one if a J-to-K transition is detected while the port is in the Suspend state. When this bit transitions to a one because a J-to-K transition is detected, the Port Change Detect bit in the USBSTS register is also set to one. This bit will automatically change to zero after the resume sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the resume duration is timed in the driver. Note that when the Host controller owns the port, the resume sequence follows the defined sequence documented in the USB Specification Revision 2.0. The resume signaling (Full-speed 'K') is driven on the port as long as this bit remains a one. This bit remains a one until the port has switched to the high-speed idle. Writing a zero has no effect because the port controller will time 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.

Bit	R/W	Reset	Description
			0 = NO_RESUME 1 = RESUME
5	RO	X	OCC: Over-current Change: Not supported 0 = NO_CHANGE 1 = CHANGE
4	RO	X	OCA: Over-current Active: Not supported 0 = NO_OVER_CURRENT 1 = OVER_CURRENT
3	RW	0x0	PEC: Port Enable/Disable Change: 1=Port enabled/disabled status has changed. 0=No change. In Host Mode: For the root hub, this bit gets set to a one only when a port is disabled due to disconnect on the port or due to the appropriate conditions existing at the EOF2 point (See Chapter 11 of the USB Specification). Software clears this by writing a one to it. This field is zero if Port Power(PP) is zero. In Device mode: The device port is always enabled. (This bit will be zero) 0 = NO_CHANGE 1 = CHANGE
2	RW	0x1	PE: Port Enabled/Disabled: 1=Enable. 0=Disable (default) In Host Mode: Ports can only be enabled by the host controller as a part of the reset and enable. Software cannot enable a port by writing a one to this field. Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by the host software. Note that the bit status does not change until the port state actually changes. There may be a delay in disabling or enabling a port due to other host controller and bus events. When the port is disabled, (0b) downstream propagation of data is blocked except for reset. This field is zero if Port Power(PP) is zero in host mode. In Device Mode: The device port is always enabled. (This bit will be one) 0 = PORT_DISABLED 1 = PORT_ENABLED
1	RW	0x0	CSC: Connect Status Change: 1 =Change in Current Connect Status. 0=No change (default) In Host Mode: Indicates a change has occurred in the port's Current Connect Status. The host/device controller sets this bit for all changes to the port device connect status, even if system software has not cleared an existing connect status change. For example, the insertion status changes twice before system software has cleared the changed condition, hub hardware will be 'setting' an already-set bit (i.e., the bit will remain set). Software clears this bit by writing a one to it. This field is zero if Port Power(PP) is zero in host mode. This bit is undefined in device controller mode. 0 = NO_CHANGE 1 = CHANGE
0	RO	X	CCS: Current Connect Status: In Host Mode: 1=Device is present on port. 0=No device is present (default) This value reflects the current state of the port, and may not correspond directly to the event that caused the Connect Status Change bit (Bit 1) to be set. This field is zero if Port Power(PP) is zero in host mode. In Device Mode: 1=Attached 0=Not Attached (default) A one indicates that the device successfully attached and is operating in either high speed or full speed as indicated by the High Speed Port bit in this register. A zero indicates that the device did not attach successfully or was forcibly disconnected by the software writing a zero to the Run bit in the USBCMD register. It does not state the device being disconnected or suspended. 0 = NOT_CONNECTED 1 = CONNECTED

26.5.1.22 USB2_CONTROLLER_USB2D_OTGSC_0

USB2D Status and Control Register

Offset: 1a4h | Read/Write: R/W | Reset: 0b0000000x0000000xxxxxxxxxxx00x00

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.

Bit	R/W	Reset	Description
			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
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

Bit	R/W	Reset	Description
			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
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: Termination. This bit must be set when the 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

26.5.1.23 USB2_CONTROLLER_USB2D_USBMODE_0

USB2D USB Device Mode Register

Offset: 1a8h | Read/Write: RO | Reset: 0bxxxxx

Bit	Reset	Description
4	X	SDIS: Stream disbable: 1 Streaming is disabled - helpful to avoid overrun/underruns when system load is too high. 0 = STREAM_ENABLE 1 = STREAM_DISABLE
3	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_OFF 1 = LOCKOUT_ON
2	X	ES: Endian Select: Note: For this implementation, this should be always set to 0 (little endian). 0 = LITTLE_ENDIAN 1 = RESERVED
1:0	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

Bit	Reset	Description
		0 = IDLE 1 = RESERVED 2 = DEVICE_MODE 3 = HOST_MODE

26.5.1.24 USB2_CONTROLLER_USB2D_ENDPTSETUPSTAT_0

USB2D Endpoint Setup Status Register

Offset: 1ach | Read/Write: R/W | Reset: 0b0000000000000000

Bit	Reset	Description
15	0x0	ENDPTSETUPSTAT15: Endpoint 15 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
14	0x0	ENDPTSETUPSTAT14: Endpoint 14 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
13	0x0	ENDPTSETUPSTAT13: Endpoint 13 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
12	0x0	ENDPTSETUPSTAT12: Endpoint 12 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
11	0x0	ENDPTSETUPSTAT11: Endpoint 11 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
10	0x0	ENDPTSETUPSTAT10: Endpoint 10 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
9	0x0	ENDPTSETUPSTAT9: Endpoint 9 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD

Bit	Reset	Description
		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
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

Bit	Reset	Description
		data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD

26.5.1.25 USB2_CONTROLLER_USB2D_ENDPTPRIME_0

USB2D Endpoint Initialization Register

Offset: 1b0h | 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 endpiont. 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 endpiont. 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 endpiont. 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 endpiont. 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 endpiont. 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 endpiont. 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 endpiont. 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.

Bit	Reset	Description
		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 endpiont. 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 endpiont. 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 endpiont. 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 endpiont. 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 endpiont. 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 endpiont. 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 endpiont. 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 endpiont. 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

Bit	Reset	Description
		operation in order to respond to a USB IN/INTERRUPT transaction on this endpiont. 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
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.

Bit	Reset	Description
		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
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

26.5.1.26 USB2_CONTROLLER_USB2D_ENDPTFLUSH_0

USB2D Endpoint De-Initialization Register

Offset: 1b4h | 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
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

Bit	Reset	Description
		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
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

Bit	Reset	Description
		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
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

Bit	Reset	Description
		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

26.5.1.27 USB2_CONTROLLER_USB2D_ENDPTSTATUS_0

USB2D Endpoint Status Register

Offset: 1b8h | 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY

Bit	Reset	Description
		1 = READY
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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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

Bit	Reset	Description
		ENDPTPRIME register and endpoint indicating ready. This delay ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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
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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by

Bit	Reset	Description
		the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY

26.5.1.28 USB2_CONTROLLER_USB2D_ENDPTCOMPLETE_0

USB2D Endpoint Complete Register

Offset: 1bch | 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

Bit	Reset	Description
		USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
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

Bit	Reset	Description
		and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
19	0x0	ETCE3: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
18	0x0	ETCE2: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
17	0x0	ETCE1: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
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

Bit	Reset	Description
10	0x0	ERCE10: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
9	0x0	ERCE9: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
8	0x0	ERCE8: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
7	0x0	ERCE7: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 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

Bit	Reset	Description
		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

26.5.1.29 USB2_CONTROLLER_USB2D_ENDPTCTRL0_0

USB2D Endpoint Control 0 Register

Offset: 1c0h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

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

26.5.1.30 USB2_CONTROLLER_USB2D_ENDPTCTRL1_0

USB2D Endpoint Control 1 Register

Offset: 1c4h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING

Bit	R/W	Reset	Description
			1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above, 0 = EP_OK 1 = EP_STALL

26.5.1.31 USB2_CONTROLLER_USB2D_ENDPTCTRL2_0

USB2D Endpoint Control 2 Register

Offset: 1c8h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE

Bit	R/W	Reset	Description
			1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

26.5.1.32 USB2_CONTROLLER_USB2D_ENDPTCTRL3_0

USB2D Endpoint Control 3 Register

Offset: 1cch | Read/Write: R/W | Reset: 0b000x00x0xxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

26.5.1.33 USB2_CONTROLLER_USB2D_ENDPTCTRL4_0

USB2D Endpoint Control 4 Register

Offset: 1d0h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

26.5.1.34 USB2_CONTROLLER_USB2D_ENDPTCTRL5_0

USB2D Endpoint Control 5 Register

Offset: 1d4h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK

Bit	R/W	Reset	Description
			1 = EP_STALL

26.5.1.35 USB2_CONTROLLER_USB2D_ENDPTCTRL6_0

USB2D Endpoint Control 6 Register

Offset: 1d8h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.

Bit	R/W	Reset	Description
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

26.5.1.36 USB2_CONTROLLER_USB2D_ENDPTCTRL7_0

USB2D Endpoint Control 7 Register

Offset: 1dch | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL

Bit	R/W	Reset	Description
			1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

26.5.1.37 USB2_CONTROLLER_USB2D_ENDPTCTRL8_0

USB2D Endpoint Control 8 Register

Offset: 1e0h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.

Bit	R/W	Reset	Description
			0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

26.5.1.38 USB2_CONTROLLER_USB2D_ENDPTCTRL9_0

USB2D Endpoint Control 9 Register

Offset: 1e4h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING

Bit	R/W	Reset	Description
			1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

26.5.1.39 USB2_CONTROLLER_USB2D_ENDPTCTRL10_0

USB2D Endpoint Control 10 Register

Offset: 1e8h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK

Bit	R/W	Reset	Description
			1 = EP_STALL

26.5.1.40 USB2_CONTROLLER_USB2D_ENDPTCTRL11_0

USB2D Endpoint Control 11 Register

Offset: 1ech | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.

Bit	R/W	Reset	Description
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

26.5.1.41 USB2_CONTROLLER_USB2D_ENDPTCTRL12_0

USB2D Endpoint Control 12 Register

Offset: 1f0h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL

Bit	R/W	Reset	Description
			1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

26.5.1.42 USB2_CONTROLLER_USB2D_ENDPTCTRL13_0

USB2D Endpoint Control 13 Register

Offset: 1f4h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.

Bit	R/W	Reset	Description
			0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

26.5.1.43 USB2_CONTROLLER_USB2D_ENDPTCTRL14_0

USB2D Endpoint Control 14 Register

Offset: 1f8h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING

Bit	R/W	Reset	Description
			1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

26.5.1.44 USB2_CONTROLLER_USB2D_ENDPTCTRL15_0

USB2D Endpoint Control 15 Register

Offset: 1fch | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE

Bit	R/W	Reset	Description
			1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

26.5.2 USB1 Controller Interface Registers

Interface registers for USB1 controller. These are used to generate the actual RTL registers for USB1 controller interface.

26.5.2.1 USB1_IF_USB_SUSP_CTRL_0

USB suspend control register. This register controls the suspend and reusume behavior of USB controller/PHY.

Offset: 400h | Read/Write: R/W | Reset: 0b000x0xx0000xx000000

Bit	R/W	Reset	Description
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	0x0	USB_SUSP_SET: Suspend Set Software must write a 1 to this bit to set the PHY into suspend mode. 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
11	RW	0x0	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

Bit	R/W	Reset	Description
9	RW	0x0	USB_PHY_CLK_VALID_INT_ENB: USB PHY clock valid interrupt enable If this bit is enabled, interrupt is generated whenever USB clocks are resumed from a suspend. 0 = DISABLE 1 = ENABLE
8	RO	0x0	USB_PHY_CLK_VALID_INT_STS: USB PHY clock valid interrupt status This bit is set whenever USB PHY clock is waked up from suspend. Software must write a 1 to clear this bit. 0 = UNSET 1 = SET
7	RO	X	USB_PHY_CLK_VALID: USB PHY clock valid status This bit indicates whether the USB PHY is generating a valid clock to the USB controller. If USB PHY clock is running, this bit is set to 1, else it is set to 0. 0 = UNSET 1 = SET
6	RO	X	USB_CLKEN: USB AHB clock enable status. Indicates whether the AHB clock to the USB controller is enabled or not. If AHB clock to USB controller is enabled, this bit is set to 1, else it is set to 0. NOTE: even when this is set to 0, all essential blocks that are required to resume USB clocks from suspend will be active and their AHB clock will not be suspended. 0 = UNSET 1 = SET
5	RW	0x0	USB_SUSP_CLR: Suspend Clear Software must write a 1 to this bit to bring the PHY out of suspend mode. This is used when the software stops the PHY clock during suspend and then wants to initiate a resume. Software should also write 0 to clear it. NOTE: It is required that software generate a positive pulse on this bit to guarantee proper operation. 0 = UNSET 1 = SET
4	RW	0x0	USB_WAKE_ON_DISCON_EN_DEV: Wake on Disconnect Enable (device mode) When enabled (1), USB device will wakeup from suspend on a disconnect event. This is only valid when USB controller is in device mode, it is not applicable when USB controller is in host mode. 0 = DISABLE 1 = ENABLE
3	RW	0x0	USB_WAKE_ON_CNNT_EN_DEV: Wake on Connect Enable (device mode) When enabled (1), USB device will wakeup from suspend on a connect event. This is only valid when USB controller is in device mode, it is not applicable when USB controller is in host mode. 0 = DISABLE 1 = ENABLE
2	RW	0x0	USB_WAKE_ON_RESUME_EN: Wake on resume enable If this bit is enabled, USB will wakeup from suspend whenever a resume event is detected on USB. This is valid for both USB device and USB host modes. 0 = DISABLE 1 = ENABLE
1	RW	0x0	USB_WAKEUP_INT_ENB: USB wakeup interrupt enable If this bit is enabled, interrupt is generated whenever USB wakeup event is generated. 0 = DISABLE 1 = ENABLE
0	RO	0x0	USB_WAKEUP_INT_STS: USB wakeup interrupt status This bit is set whenever USB wakes up from suspend (a wakeup event is generated). Software must write a 1 to clear this bit. Note that during the wakeup sequence, PHY clocks will be resumed from suspend. Software can check when the PHY clocks are resumed by reading the bit USB_PHY_CLK_VALID. There is also a separate interrupt generated when PHY clock is resumed if USB_PHY_CLK_VALID_INT_EN is set. During the wakeup sequence, first USB_WAKEUP_INT_STS will be set, and it will take some time for the PHY clock to resume, which can be detected by checking USB_PHY_CLK_VALID. 0 = UNSET 1 = SET

26.5.2.2 USB1_IF_USB_PHY_VBUS_SENSORS_0

USB PHY VBUS SENSORS control register

This register controls the VBUS sensors in the USB PHY.

We have the following 4 VBUS sensors:

1. A_VBUS_VLD

2. A_SESS_VLD

3. B_SESS_VLD

4. B_SESS_END

The debounced status of each sensor can be read from _STS. The field

_CHG_DET is set to 1 whenever a change is detected in the value of

. If _INT_EN is set, then an interrupt is generated to the

processor. This interrupt can be routed to CPU/COP by appropriately writing the USBD

bits in the interrupt controller registers.

In case Software wants to override the value for a , it can set

_SW_EN to 1, and set _SW_VALUE to 1 or 0 as per the requirement.

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 apprORpriate value (SEL_A or

SEL_B).

NOTE1: 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.

NOTE2: The source for A_SESS_VLD sensor can be programmed according to the setting of

USB1_VBUS_SENSE_CTL in register USB1_LEGACY_CTRL.

Offset: 404h | Read/Write: R/W | Reset: 0b0000x00x0000x00x0000x00x0000x00

Bit	R/W	Reset	Description
30	RW	0x0	A_VBUS_VLD_WAKEUP_EN: A_VBUS_VLD wakeup enable If this bit is enabled, USB will wakeup from suspend whenever a change is detected on A_VBUS_VLD. 0 = DISABLE 1 = ENABLE
29	RW	0x0	A_VBUS_VLD_DEB_SEL_B: A_VBUS_VLD debounce A/B select Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
28	RW	0x0	A_VBUS_VLD_SW_VALUE: A_VBUS_VLD software value Software should write the appropriate value (1/0) to set/unset the A_VBUS_VLD status. This is only valid when A_VBUS_VLD_SW_EN is set. 0 = UNSET 1 = SET
27	RW	0x0	A_VBUS_VLD_SW_EN: A_VBUS_VLD software enable Enable Software Controlled A_VBUS_VLD. Software sets this bit to drive the value in A_VBUS_VLD_SW_VALUE to the USB controller. 0 = DISABLE

Bit	R/W	Reset	Description
			1 = ENABLE
26	RO	X	A_VBUS_VLD_STS: A_VBUS_VLD status This is set to 1 whenever A_VBUS_VLD sensor output is 1. 0 = UNSET 1 = SET
25	RO	0x0	A_VBUS_VLD_CHG_DET: A_VBUS_VLD change detect. This field is set by hardware whenever a change is detected in the value of A_VBUS_VLD. software writes a 1 to clear it 0 = UNSET 1 = SET
24	RW	0x0	A_VBUS_VLD_INT_EN: A_VBUS_VLD interrupt enable If this field is set to 1, an interrupt is generated whenever A_VBUS_VLD_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
22	RW	0x0	A_SESS_VLD_WAKEUP_EN: A_SESS_VLD wakeup enable If this bit is enabled, USB will wakeup from suspend whenever a change is detected on A_SESS_VLD. 0 = DISABLE 1 = ENABLE
21	RW	0x0	A_SESS_VLD_DEB_SEL_B: A_SESS_VLD debounce A/B select Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
20	RW	0x0	A_SESS_VLD_SW_VALUE: A_SESS_VLD software value Software should write the appropriate value (1/0) to set/unset the A_SESS_VLD status. This is only valid when A_SESS_VLD_SW_EN is set. 0 = UNSET 1 = SET
19	RW	0x0	A_SESS_VLD_SW_EN: A_SESS_VLD software enable Enable Software Controlled A_SESS_VLD. Software sets this bit to drive the value in A_SESS_VLD_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
18	RO	X	A_SESS_VLD_STS: A_SESS_VLD status This is set to 1 whenever A_SESS_VLD sensor output is 1. 0 = UNSET 1 = SET
17	RO	0x0	A_SESS_VLD_CHG_DET: A_SESS_VLD change detect. This field is set by hardware whenever a change is detected in the value of A_SESS_VLD. software writes a 1 to clear it 0 = UNSET 1 = SET
16	RW	0x0	A_SESS_VLD_INT_EN: A_SESS_VLD interrupt enable If this field is set to 1, an interrupt is generated whenever A_SESS_VLD_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
14	RW	0x0	B_SESS_VLD_WAKEUP_EN: B_SESS_VLD wakeup enable If this bit is enabled, USB will wakeup from suspend whenever a change is detected on B_SESS_VLD. 0 = DISABLE 1 = ENABLE
13	RW	0x0	B_SESS_VLD_DEB_SEL_B: B_SESS_VLD debounce A/B select Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
12	RW	0x0	B_SESS_VLD_SW_VALUE: B_SESS_VLD software value Software should write the appropriate value (1/0) to set/unset the B_SESS_VLD status. This is only valid when B_SESS_VLD_SW_EN is set.

Bit	R/W	Reset	Description
			0 = UNSET 1 = SET
11	RW	0x0	B_SESS_VLD_SW_EN: B_SESS_VLD software enable Enable Software Controlled B_SESS_VLD. Software sets this bit to drive the value in B_SESS_VLD_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
10	RO	X	B_SESS_VLD_STS: B_SESS_VLD status This is set to 1 whenever B_SESS_VLD sensor output is 1. 0 = UNSET 1 = SET
9	RO	0x0	B_SESS_VLD_CHG_DET: B_SESS_VLD change detect. This field is set by hardware whenever a change is detected in the value of B_SESS_VLD. software writes a 1 to clear it 0 = UNSET 1 = SET
8	RW	0x0	B_SESS_VLD_INT_EN: B_SESS_VLD interrupt enable If this field is set to 1, an interrupt is generated whenever B_SESS_VLD_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
6	RW	0x0	B_SESS_END_WAKEUP_EN: B_SESS_END wakeup enable If this bit is enabled, USB will wakeup from suspend whenever a change is detected on B_SESS_END. 0 = DISABLE 1 = ENABLE
5	RW	0x0	B_SESS_END_DEB_SEL_B: B_SESS_END debounce A/B select Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
4	RW	0x0	B_SESS_END_SW_VALUE: B_SESS_END software value Software should write the appropriate value (1/0) to set/unset the B_SESS_END status. This is only valid when B_SESS_END_SW_EN is set. 0 = UNSET 1 = SET
3	RW	0x0	B_SESS_END_SW_EN: B_SESS_END software enable Enable Software Controlled B_SESS_END Software sets this bit to drive the value in B_SESS_END_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
2	RO	X	B_SESS_END_STS: B_SESS_END status This is set to 1 whenever B_SESS_END sensor output is 1. 0 = UNSET 1 = SET
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

26.5.2.3 USB1_IF_USB_PHY_VBUS_WAKEUP_ID_0

USB PHY VBUS wakeup and ID control register

This register controls the battery charger (VDAT_DET), VBUS_WAKEUP and ID sensors.

The following sensors are in this register:

1. VBUS_WAKEUP
2. ID
3. VDAT_DET

The debounced status of each sensor can be read from _STS. The field _CHG_DET is set to 1 whenever a change is detected in the value of If _INT_EN is set, then an interrupt is generated to the processor. This interrupt can be routed to CPU/COP by appropriately writing the USBD bits in the interrupt controller registers.

In case Software wants to override the value for a , it can set _SW_EN to 1, and set _SW_VALUE to 1 or 0 as per the requirement.

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 approrpriate value (SEL_A or SEL_B).

NOTE: Do not set either UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B to 0x0. If not using one of the debouncers, keep it at the default value of 0xFFFF.

There is no debouncer for VDAT_DET.

Offset: 408h | Read/Write: R/W | Reset: 0b0xxxxxxxx000x00xx000x00x1000x00

Bit	R/W	Reset	Description
30	RW	0x0	VBUS_WAKEUP_WAKEUP_EN: VBUS_WAKEUP wakeup enable If this bit is enabled, USB will wakeup from suspend whenever a change is detected on VBUS_WAKEUP. 0 = DISABLE 1 = ENABLE
21	RW	0x0	VDAT_DET_DEB_SEL_B: VDAT_DET 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	VDAT_DET_SW_VALUE: VDAT_DET software value Software should write the appropriate value (1/0) to set/unset the VDAT_DET status. This is only valid when VDAT_DET_SW_EN is set. 0 = UNSET 1 = SET
19	RW	0x0	VDAT_DET_SW_EN: VDAT_DET software enable Enable Software Controlled VDAT_DET. Software sets this bit to drive the value in VDAT_DET_SW_VALUE to the USB controller 0 = DISABLE 1 = ENABLE
18	RO	X	VDAT_DET_STS: VDAT_DET status This is set to 1 whenever VDAT_DET sensor output is 1. 0 = UNSET 1 = SET
17	RO	0x0	VDAT_DET_CHG_DET: VDAT_DET change detect. This field is set by hardware whenever a change is detected in the value of VDAT_DET. software writes a 1 to clear it 0 = UNSET 1 = SET
16	RW	0x0	VDAT_DET_INT_EN: VDAT_DET interrupt enable If this field is set to 1, an interrupt is generated whenever VDAT_DET_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
13	RW	0x0	VBUS_WAKEUP_DEB_SEL_B: VBUS_WAKEUP debounce A/B select Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register

Bit	R/W	Reset	Description
			UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
12	RW	0x0	VBUS_WAKEUP_SW_VALUE: VBUS wakeup software value Software should write the appropriate value (1/0) to set/unset the VBUS_WAKEUP status. This is only valid when VBUS_WAKEUP_SW_EN is set. 0 = UNSET 1 = SET
11	RW	0x0	VBUS_WAKEUP_SW_EN: VBUS wakeup software enable Enable Software Controlled VBUS_WAKEUP. Software sets this bit to drive the value in VBUS_WAKEUP_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
10	RO	X	VBUS_WAKEUP_STS: VBUS wakeup status This is set to 1 whenever VBUS_WAKEUP sensor output is 1. 0 = UNSET 1 = SET
9	RO	0x0	VBUS_WAKEUP_CHG_DET: VBUS wakeup change detect. This field is set by hardware whenever a change is detected in the value of VBUS_WAKEUP. software writes a 1 to clear it 0 = UNSET 1 = SET
8	RW	0x0	VBUS_WAKEUP_INT_EN: VBUS wakeup interrupt enable If this field is set to 1, an interrupt is generated whenever VBUS_WAKEUP_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
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

26.5.2.4 USB1_IF_USB_PHY_ALT_VBUS_STS_0

USB PHY Alternate VBUS status register

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

Bit	Reset	Description
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

26.5.2.5 USB1_IF_USB1_LEGACY_CTRL_0

USB1 Legacy control register.

This register controls legacy mode access for USB controller.

Offset: 410h | Read/Write: R/W | Reset: 0b000

Bit	Reset	Description
2:1	0x0	USB1_VBUS_SENSE_CTL: Vbus_sense control Controls which VBUS sensor input is driven to the controller. 00: Use VBUS_WAKEUP. 01: Use (A_SESS_VLD || B_SESS_VLD) output from the PHY if the PHY clock is available. Otherwise, use VBUS_WAKEUP. 10: Use (A_SESS_VLD || B_SESS_VLD) output from the PHY 11: Use A_SESS_VLD output from the PHY 0 = VBUS_WAKEUP 1 = AB_SESS_VLD_OR_VBUS_WAKEUP 2 = AB_SESS_VLD 3 = A_SESS_VLD
0	0x0	USB1_NO_LEGACY_MODE: Legacy registers select The default is to select legacy mode registers in APB_MISC for USB1. Selects new registers if this is set to 1. 0 = LEGACY 1 = NEW

26.5.2.6 USB1_IF_USB_INTER_PKT_DELAY_CTRL_0

Inter packet delay control. This controls the interpacket delay between two consecutive transmit packets in HS mode. This only applies to host mode as device never transmits two packets in a row.

Offset: 420h | Read/Write: R/W | Reset: 0b000110

Bit	Reset	Description
5:0	0x6	IP_DELAY_TX2TX_HS: HS Tx to Tx inter-packet delay. Software should not change this.

26.5.2.7 USB2_QH_USB2D_QH_EP_0_OUT_0

USB2D Queue Head for OUT endpoint 0

Offset: 1000h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 0 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 0.

26.5.2.8 USB2_QH_USB2D_QH_EP_0_IN_0

USB2D Queue Head for IN endpoint 0

Offset: 1040h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 0. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 0.

26.5.2.9 USB2_QH_USB2D_QH_EP_1_OUT_0

USB2D Queue Head for OUT endpoint 1

Offset: 1080h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 1 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 1.

26.5.2.10 USB2_QH_USB2D_QH_EP_1_IN_0

USB2D Queue Head for IN endpoint 1

Offset: 10c0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 1. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 1.

26.5.2.11 USB2_QH_USB2D_QH_EP_2_OUT_0

USB2D Queue Head for OUT endpoint 2

Offset: 1100h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 2 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 2.

26.5.2.12 USB2_QH_USB2D_QH_EP_2_IN_0

USB2D Queue Head for IN endpoint 2

Offset: 1140h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 2. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 2.

26.5.2.13 USB2_QH_USB2D_QH_EP_3_OUT_0

USB2D Queue Head for OUT endpoint 3

Offset: 1180h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 3 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 3.

26.5.2.14 USB2_QH_USB2D_QH_EP_3_IN_0

USB2D Queue Head for IN endpoint 3

Offset: 11c0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 3. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 3.

26.5.2.15 USB2_QH_USB2D_QH_EP_4_OUT_0

USB2D Queue Head for OUT endpoint 4

Offset: 1200h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 4 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 4.

26.5.2.16 USB2_QH_USB2D_QH_EP_4_IN_0

USB2D Queue Head for IN endpoint 4

Offset: 1240h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 4. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 4.

26.5.2.17 USB2_QH_USB2D_QH_EP_5_OUT_0

USB2D Queue Head for OUT endpoint 5

Offset: 1280h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 5 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 5.

26.5.2.18 USB2_QH_USB2D_QH_EP_5_IN_0

USB2D Queue Head for IN endpoint 5

Offset: 12c0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 5. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 5.

26.5.2.19 USB2_QH_USB2D_QH_EP_6_OUT_0

USB2D Queue Head for OUT endpoint 6

Offset: 1300h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 6 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 6.

26.5.2.20 USB2_QH_USB2D_QH_EP_6_IN_0

USB2D Queue Head for IN endpoint 6

Offset: 1340h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 6. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 6.

26.5.2.21 USB2_QH_USB2D_QH_EP_7_OUT_0

USB2D Queue Head for OUT endpoint 7

Offset: 1380h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 7 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 7.

26.5.2.22 USB2_QH_USB2D_QH_EP_7_IN_0

USB2D Queue Head for IN endpoint 7

Offset: 13c0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 7. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 7.

26.5.2.23 USB2_QH_USB2D_QH_EP_8_OUT_0

USB2D Queue Head for OUT endpoint 8

Offset: 1400h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 8 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 8.

26.5.2.24 USB2_QH_USB2D_QH_EP_8_IN_0

USB2D Queue Head for IN endpoint 8

Offset: 1440h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 8. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 8.

26.5.2.25 USB2_QH_USB2D_QH_EP_9_OUT_0

USB2D Queue Head for OUT endpoint 9

Offset: 1480h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 9 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 9.

26.5.2.26 USB2_QH_USB2D_QH_EP_9_IN_0

USB2D Queue Head for IN endpoint 9

Offset: 14c0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 9. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 9.

26.5.2.27 USB2_QH_USB2D_QH_EP_10_OUT_0

USB2D Queue Head for OUT endpoint 10

Offset: 1500h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 10 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 10.

26.5.2.28 USB2_QH_USB2D_QH_EP_10_IN_0

USB2D Queue Head for IN endpoint 10

Offset: 1540h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 10. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 10.

26.5.2.29 USB2_QH_USB2D_QH_EP_11_OUT_0

USB2D Queue Head for OUT endpoint 11

Offset: 1580h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 11 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 11.

26.5.2.30 USB2_QH_USB2D_QH_EP_11_IN_0

USB2D Queue Head for IN endpoint 11

Offset: 15c0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 11. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 11.

26.5.2.31 USB2_QH_USB2D_QH_EP_12_OUT_0

USB2D Queue Head for OUT endpoint 12

Offset: 1600h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 12 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 12.

26.5.2.32 USB2_QH_USB2D_QH_EP_12_IN_0

USB2D Queue Head for IN endpoint 12

Offset: 1640h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 12. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 12.

26.5.2.33 USB2_QH_USB2D_QH_EP_13_OUT_0

USB2D Queue Head for OUT endpoint 13

Offset: 1680h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 13 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 13.

26.5.2.34 USB2_QH_USB2D_QH_EP_13_IN_0

USB2D Queue Head for IN endpoint 13

Offset: 16c0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 13. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 13.

26.5.2.35 USB2_QH_USB2D_QH_EP_14_OUT_0

USB2D Queue Head for OUT endpoint 14

Offset: 1700h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 14 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 14.

26.5.2.36 USB2_QH_USB2D_QH_EP_14_IN_0

USB2D Queue Head for IN endpoint 14

Offset: 1740h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 14. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 14.

26.5.2.37 USB2_QH_USB2D_QH_EP_15_OUT_0

USB2D Queue Head for OUT endpoint 15

Offset: 1780h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 15 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 15.

26.5.2.38 USB2_QH_USB2D_QH_EP_15_IN_0

USB2D Queue Head for IN endpoint 15

Offset: 17c0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 15. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 15.

26.5.2.39 USB1_UTMIP_PLL_CFG0_0

USB_PHY PLL Configuration Register 0 The data sampling frequency relies on a 960MHz clock, so the goal of the PLL is to have: In_Frequency*(PLL_VCOMULTBY2+1)*(PLL_NDIV/PLL_MDIV) = 960MHz With a 12MHz input from PLL_U, the default setting of PLL_VCOMULTBY2=1, PLL_NDIV=40 and PLL_MDIV=1 results in a correct output.

Offset: 800h | Read/Write: R/W | Reset: 0b00000000010100000000000110000000

Bit	Reset	Description
30:28	0x0	UTMIP_PLL_SELECT: Selects which of the eight 60MHz clock phases to produce at the output of the USB PHY PLL. This is for a test mode. See cell specification.
27	0x0	UTMIP_PLL_PDIVRST: PDIVRST of the UTMIP PHY PLL. Reserved. Keep at 0x0. Currently there is no post divider on this PLL. See cell specification.
26:24	0x0	UTMIP_PLL_PDIV: PDIV[2:0] input of the USB_PHY PLL. Reserved. Keep at 0x0. Currently there is no post divider on this PLL. See cell specification.
23:16	0x28	UTMIP_PLL_NDIV: NDIV[7:0] input of USB_PHY PLL. This is the feedback divider on the VCO feedback. 0x0 is not allowed. See cell specification.
15:8	0x1	UTMIP_PLL_MDIV: MDIV[7:0] input of the USB_PHY PLL. This is the predivide on the PLL. 0x0 is not allowed. See cell specification.
7	0x1	UTMIP_PLL_VCOMULTBY2: VCOMULTBY2 control of the UTMIP PHY PLL. Recommended setting is on. Additional divide by 2 on the VCO feedback. Which is setting the bit to 1. See cell spec.
6:1	0x0	UTMIP_PLL_LOCKSEL: LOCKSEL[5:0] input of USB_PHY PLL. Used in test modes only. See cell specification.
0	0x0	UTMIP_PLL_LOCKENABLE: LOCK_ENABLE input of USB_PHY PLL. Normally only used during test. See cell specification.

26.5.2.40 USB1_UTMIP_PLL_CFG1_0

UTMIP PLL and PLLU configuration register 1

PLL CONFIGURATION & PARAMETERS

In normal operation, the following clock generators are in play for USB: Xtal clock -> enters PLL_U to generate 12MHz clock -> enters USB_PHY PLL to generate 480/60 MHz clock

The following parameters control the bringup of the plls:

Coming out of reset or suspend

-> PllUOnState: start pllu_enable_count and pll_lock_count

-> Wait ~1us to actually enable the PLL_U (pllu_enable_count == ClkXtal * PLLU_ENABLE_DLY_COUNT * 8)

-> Wait ~1ms until PLL_U is actually stable (pll_lock_count == ClkXtal * PLLU_STABLE_COUNT * 256) => USB_PHY PLL_ENABLE

pll_active_count = 0

-> Wait 5us to enable pll_active: pll_active_count == ClkXtal * PLL_ACTIVE_DLY_COUNT * 16 => USB_PHY PLL_ACTIVE

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

Numbers for a 19.2MHz Xtal clock

PLLU_ENABLE_DLY_COUNT[4:0] = 1 * 19.2 / 8 = 2.4 = 3 = 0x03

PLLU_STABLE_COUNT[11:0] = 1000 * 19.2 / 256 = 75 = 0x4b

PLL_ACTIVE_DLY_COUNT[4:0] = 5 * 19.2 / 16 = 6 = 0x06

XTAL_FREQ_COUNT[11:0] = 2500 * 19.2 / 256 = 187 = 0xbb

Offset: 804h | Read/Write: R/W | Reset: 0b00011000001000000000000011000000

Bit	Reset	Description
31:27	0x3	UTMIP_PLLU_ENABLE_DLY_COUNT: Controls the wait time to enable PLL_U when coming out of suspend or reset.
26:18	0x8	UTMIP_PLL_SETUP: SETUP[8:0] input of USB_PHY PLL.
17	0x0	UTMIP_FORCE_PLLU_POWERUP: Force PLL_U into power up.
16	0x0	UTMIP_FORCE_PLLU_POWERDOWN: Force PLL_U into power down. (Overrides FORCE_PLLU_POWERUP.)
15	0x0	UTMIP_FORCE_PLL_ENABLE_POWERUP: Force USB_PHY PLL pll_enable input on.
14	0x0	UTMIP_FORCE_PLL_ENABLE_POWERDOWN: Force USB_PHY PLL pll_enable input off. (Overrides FORCE_PLL_ENABLE_POWERUP.)
13	0x0	UTMIP_FORCE_PLL_ACTIVE_POWERUP: Force USB_PHY PLL pll_active input on.
12	0x0	UTMIP_FORCE_PLL_ACTIVE_POWERDOWN: Force USB_PHY PLL pll_active input off. (Overrides FORCE_PLL_ACTIVE_POWERUP.)
11:0	0xc0	UTMIP_XTAL_FREQ_COUNT: Determines the time to wait until the output of USB_PHY PLL is considered stable.

26.5.2.41 USB1_UTMIP_XCVR_CFG0_0

UTMIP transceiver cell configuration register 0

UTMIP REGISTER: UTMIP_XCVR_CFG0

Define spec-dependent variable

Offset: 808h | Read/Write: R/W | Reset: 0b00100000001000000010010100000000

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	0x0	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	0x0	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	0x0	UTMIP_FORCE_PD_POWERDOWN: Force PD input into power down. (Overrides FORCE_PD_POWERUP.)
13	0x1	UTMIP_XCVR_TERMEN: Enable HS termination.
12	0x0	UTMIP_XCVR_HSLOOPBACK: Internal loopback inside XCVR cell. Used for IOBIST.
11:10	0x1	UTMIP_XCVR_LSFSLEW: LS falling slew rate control.
9:8	0x1	UTMIP_XCVR_LSRSLEW: LS rising slew rate control.

Bit	Reset	Description
7:6	0x0	UTMIP_XCVR_FSSLEW: FS slew rate control.
5:4	0x0	UTMIP_XCVR_HSSLEW: HS slew rate control. The two LSBs.
3:0	0x0	UTMIP_XCVR_SETUP: SETUP[3:0] input of XCVR cell. HS driver output control. 4 LSBs.

26.5.2.42 USB1_UTMIP_BIAS_CFG0_0

UTMIP Bias cell configuration register 0

UTMIP REGISTER: UTMIP_BIAS_CFG0

Offset: 80ch | Read/Write: R/W | Reset: 0b00000000000000000000000000

Bit	Reset	Description
24	0x0	UTMIP_HSDISCON_LEVEL_MSB: Most significant bit of UTMIP_HSDISCON_LEVEL, bit 2
23	0x0	UTMIP_IDPD_VAL: See IDPD_SEL.
22	0x0	UTMIP_IDPD_SEL: 0: IDPD = ~IdPullup. 1: IDPD = IDPD_VAL.
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	0x0	UTMIP_OTGPD: Power down circuit.
10	0x0	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.

26.5.2.43 USB1_UTMIP_HSRX_CFG0_0

UTMIP High speed receive config 0

UTMIP REGISTER: UTMIP_HSRX_CFG0

Offset: 810h | Read/Write: R/W | Reset: 0b10010001011001010011010000000000

Bit	Reset	Description
31:30	0x2	UTMIP_KEEP_PATT_ON_ACTIVE: Keep the stay alive pattern on active
29	0x0	UTMIP_ALLOW_CONSEC_UPDN: Allow consecutive ups and downs on the bits, debug only, set to 0.
28	0x1	UTMIP_REALIGN_ON_NEW_PKT: Realign the inertia counters on a new packet
27:24	0x1	UTMIP_PCOUNT_UPDN_DIV: The number of (edges-1) needed to move the sampling point
23:21	0x3	UTMIP_SQUELCH_EOP_DLY: Limit the delay of the squelch at EOP time
20	0x0	UTMIP_NO_STRIPPING: Do not strip incoming data
19:15	0xa	UTMIP_IDLE_WAIT: Number of cycles of idle to declare IDLE.
14:10	0xd	UTMIP_ELASTIC_LIMIT: Depth of elastic input store
9	0x0	UTMIP_ELASTIC_OVERRUN_DISABLE: Do not declare overrun errors until overflow of FIFO
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

26.5.2.44 USB1_UTMIP_HSRX_CFG1_0

UTMIP High speed receive config 1

UTMIP REGISTER: UTMIP_HSRX_CFG1

Offset: 814h | Read/Write: R/W | Reset: 0b010011

Bit	Reset	Description
5:1	0x9	UTMIP_HS_SYNC_START_DLY: How long to wait before start of sync launches RxActive
0	0x1	UTMIP_HS_ALLOW_KEEP_ALIVE: Allow Keep Alive packets

26.5.2.45 USB1_UTMIP_FSLSRX_CFG0_0

UTMIP full and Low speed receive config 0

UTMIP REGISTER: UTMIP_FSLSRX_CFG0

Offset: 818h | Read/Write: R/W | Reset: 0b11111101010101001000010000101001

Bit	Reset	Description
31	0x1	UTMIP_FSLS_SE1_DRIBBLE_FILTER: One SE1, don't allow dribble
30	0x1	UTMIP_FSLS_SE1_FILTER: Filter SE1
29	0x1	UTMIP_FSLS_SERIAL_SE0_RCV
28:26	0x7	UTMIP_FSLS_UPR_DRIBBLE_SIZE: Do not allow <= dribble bits
25:23	0x2	UTMIP_FSLS_LWR_DRIBBLE_SIZE: Do not allow >= dribble bits
22	0x1	UTMIP_FSLS_EOP_ENDS_AT_SE0: Only look for transitioning out of EOP
21:16	0x14	UTMIP_FSLS_KCOUNT_MAX: Number of K bits in question
15	0x1	UTMIP_FSLS_KCOUNT_LIMIT: Limit the number of bit times a K can last
14	0x0	UTMIP_FSLS_ACTIVE_ON_FULL_SYNC: Require a full sync pattern to declare the data received
13:8	0x4	UTMIP_FSLS_IDLE_WAIT_MAX: 4 bits of of SEO should exceed the time limit
7	0x0	UTMIP_FSLS_IDLE_WAIT_LIMIT: Enable the reset of the state machine on extended SE0
6:1	0x14	UTMIP_FSLS_IDLE_COUNT_MAX: 20 bits of idle should end the packet if FsLsIdleCountLimitCfg=1.
0	0x1	UTMIP_FSLS_IDLE_COUNT_LIMIT: Give up on packet if a long sequence of J

26.5.2.46 USB1_UTMIP_FSLSRX_CFG1_0

UTMIP full and Low speed receive config 1

Offset: 81ch | Read/Write: R/W | Reset: 0b01000100110011101010000000000

Bit	Reset	Description
26	0x0	UTMIP_EARLY_LINE_STATE_FILTER: Assumes line state filtering table is inclusive, not exclusive
25:23	0x4	UTMIP_LS_BOUNCE_LENGTH: Number of clock cycle of LS stable
22:17	0x13	UTMIP_LS_EXTRACTION_COUNT: Phase count on which LS bits are extracted
16:11	0xe	UTMIP_LS_EOP_START_COUNT: Number of SEO clock cycles to block bit extraction
10:5	0x20	UTMIP_LS_SE0_COUNT: Only for this number of 60MHz of SEO and Idle to end packet
4	0x0	UTMIP_LS_LENIENT_DRIBBLE: Allow for large dribble in low speed mode
3	0x0	UTMIP_FS_LENIENT_DRIBBLE: Allow for large dribble in full speed mode
2	0x0	UTMIP_FS_WEAK_SYNC: Only look for a KK pattern, instead of KJKK
1	0x0	UTMIP_FS_DEBOUNCE: Whether full speed uses debouncing
0	0x0	UTMIP_FS_EOP_LENGTH: Whether full speed EOP is determined within 3(0) or 4(1) 60MHz cycles

26.5.2.47 USB1_UTMIP_TX_CFG0_0

UTMIP transmit config signals

UTMIP REGISTER: UTMIP_TX_CFG0

Offset: 820h | Read/Write: R/W | Reset: 0b00010000001000000000

Bit	Reset	Description
19	0x0	UTMIP_FS_PREAMBLE_J: output enable sends an initial J before sync pattern
18	0x0	UTMIP_FS_POSTAMBLE_OUTPUT_ENABLE: output enable turns off 1/2 cycle after
17	0x0	UTMIP_FS_PREAMBLE_OUTPUT_ENABLE: output enable turns on 1/2 cycle before
16	0x1	UTMIP_FSLS_ALLOW_SOP_TX_STUFF_ERR: Allow SOP to be source of transmit error stuffing
15	0x0	UTMIP_HS_READY_WAIT_FOR_VALID
14:10	0x0	UTMIP_HS_TX_IPG_DLY
9	0x1	UTMIP_HS_DISCON_EOP_ONLY: Only check during EOP
8	0x0	UTMIP_HS_DISCON_DISABLE: Disable high speed disconnect
7	0x0	UTMIP_HS_POSTAMBLE_OUTPUT_ENABLE: output enable turns off 1 cycle after
6	0x0	UTMIP_HS_PREAMBLE_OUTPUT_ENABLE: output enable turns on 1 cycle before
5	0x0	UTMIP_SIE_RESUME_ON_LINESTATE: SIE, not macrocell, detects LineState change to resume
4	0x0	UTMIP_SOF_ON_NO_STUFF: Sof when OpMode 3 -- perhaps, when sending controller made packets
3	0x0	UTMIP_SOF_ON_NO_ENCODE: Sof when OpMode 2 -- not likely, for Chirp
2	0x0	UTMIP_NO_STUFFING: No bit stuffing, static programming
1	0x0	UTMIP_NO_ENCODING: No encoding, static programming
0	0x0	UTMIP_NO_SYNC_NO_EOP: Do not sent SYNC or EOP

26.5.2.48 USB1_UTMIP_MISC_CFG0_0

UTMIP miscellaneous configurations

UTMIP REGISTER: UTMIP_MISC_CFG0

Offset: 824h | Read/Write: R/W | Reset: 0b00000111110000000000000001111000

Bit	Reset	Description
30:27	0x0	UTMIP_DPDM_OBSERVE_SEL: Select DP/DM obs signals
26	0x0	UTMIP_DPDM_OBSERVE: Use DP/DM as obs bus
25	0x1	UTMIP_KEEP_XCVR_PD_ON_SOFT_DISCON
24	0x1	UTMIP_ALLOW_LS_ON_SOFT_DISCON
23	0x1	UTMIP_FORCE_FS_DISABLE_ON_DEV_CHIRP
22	0x1	UTMIP_SUSPEND_EXIT_ON_EDGE: Suspend exit requires edge or simply a value...
21	0x1	UTMIP_LS_TO_FS_SKIP_4MS: Don't block changes for 4ms when going from LS to FS (should not happen)

Bit	Reset	Description
20:19	0x0	UTMIP_INJECT_ERROR_TYPE: Force error insertion into RX path. (Used for IOBIST.) 0 = DISABLE 1 = BIT_ERR 2 = RX_ERR 3 = BIT_RX_ERR
18	0x0	UTMIP_FORCE_HS_CLOCK_ON: Force HS clock always on.
17	0x0	UTMIP_DISABLE_HS_TERM: Force HS termination inactive.
16	0x0	UTMIP_FORCE_HS_TERM: Force HS termination active.
15	0x0	UTMIP_DISABLE_PULLUP_DP: Force DP pullup inactive. (Overrides FORCE_PULLUP_DP.)
14	0x0	UTMIP_DISABLE_PULLUP_DM: Force DM pullup inactive. (Overrides FORCE_PULLUP_DM.)
13	0x0	UTMIP_DISABLE_PULLDN_DP: Force DP pulldown inactive. (Overrides FORCE_PULLDN_DP.)
12	0x0	UTMIP_DISABLE_PULLDN_DM: Force DM pulldown inactive. (Overrides FORCE_PULLDN_DM.)
11	0x0	UTMIP_FORCE_PULLUP_DP: Force DP pullup active.
10	0x0	UTMIP_FORCE_PULLUP_DM: Force DM pullup active.
9	0x0	UTMIP_FORCE_PULLDN_DP: Force DP pulldown active.
8	0x0	UTMIP_FORCE_PULLDN_DM: Force DM pulldown active.
7:5	0x3	UTMIP_STABLE_COUNT: Number of cycles of crystal clock of signal not changing to consider stable.
4	0x1	UTMIP_STABLE_ALL: Determines if all signal need to be stable to not change a config.
3	0x1	UTMIP_NO_FREE_ON_SUSPEND: Don't use free running terminations during suspend.
2	0x0	UTMIP_NEVER_FREE_RUNNING_TERMS: Ignore free running terminations, even when no clock
1	0x0	UTMIP_ALWAYS_FREE_RUNNING_TERMS: Use free running terminations at all time
0	0x0	UTMIP_COMB_TERMS: Use combinational terminations or synced through CLKXTAL

26.5.2.49 USB1_UTMIP_MISC_CFG1_0

UTMIP miscellaneous configurations

UTMIP REGISTER: UTMIP_MISC_CFG1

Offset: 828h | Read/Write: R/W | Reset: 0b10000000001100110000000000100100

Bit	Reset	Description
30	0x1	UTMIP_PHY_XTAL_CLOCKEN: Selects whether to enable the crystal clock in the module.
29	0x0	UTMIP_LINESTATE_BYPASS: Bypass LineState reclocking logic
28	0x0	UTMIP_LINESTATE_NEG: Use neg edge sync for linestate
27	0x0	UTMIP_LINESTATE_XCVRSEL3: 0: Use FS filtering on line state when XcvrSel=3 1: Use LS filtering on line state when XcvrSel=3
26:25	0x0	UTMIP_OBS_SEL
24	0x0	UTMIP_FSLS_TDM
23	0x0	UTMIP_FORCE_IOBIST_CLK_ON

Bit	Reset	Description
22:18	0x6	UTMIP_PLL_ACTIVE_DLY_COUNT: 5 us / (1/19.2MHz) = 96 / 16 = 6
17:6	0x600	UTMIP_PLLU_STABLE_COUNT: WRONG! This should be 1ms -> 0x50
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

26.5.2.50 USB1_UTMIP_DEBOUNCE_CFG0_0

UTMIP Avalid and Bvalid debounce

UTMIP REGISTER: UTMIP_DEBOUNCE_CFG0

Debounce values IdDig, Avalid, Bvalid, VbusValid, VbusWakeUp, and SessEnd.

Each of these signals have their own debouncer and for each of those one out of

2 debouncing times can be chosen (BIAS_DEBOUNCE_A or BIAS_DEBOUNCE_B.)

The values of DEBOUNCE_A and DEBOUNCE_B are calculated as follows:

0xffff -> No debouncing at all

ms = *1000 / (1/19.2MHz) / 4

So to program a 1 ms debounce for BIAS_DEBOUNCE_A, we have:

BIAS_DEBOUNCE_A[15:0] = 1000 * 19.2 / 4 = 4800 = 0x12c0

Offset: 82ch | Read/Write: R/W | Reset: 0b11111111111111111111111111111111

Bit	Reset	Description
31:16	0xffff	UTMIP_BIAS_DEBOUNCE_B: simulation value -- Used for interrupts
15:0	0xffff	UTMIP_BIAS_DEBOUNCE_A: simulation value -- Used for interrupts

26.5.2.51 USB1_UTMIP_BAT_CHRG_CFG0_0

UTMIP battery charger configuration

UTMIP REGISTER: UTMIP_BAT_CHRG_CFG0

Offset: 830h | Read/Write: R/W | Reset: 0b00000

Bit	Reset	Description
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	0x0	UTMIP_PD_CHRG: Power down charger circuit

26.5.2.52 USB1_UTMIP_SPARE_CFG0_0

Utmip spare configuration bits

UTMIP REGISTER: UTMIP_SPARE_CFG0

Offset: 834h | Read/Write: R/W | Reset: 0b11111111111111110000000000000000

Bit	Reset	Description
31:0	-65536	UTMIP_SPARE: Spare register bits 0: HS_RX_IPG_ERROR_ENABLE 1: HS_RX_FLUSH_ALAP. Flush as late as possible 2: HS_RX_LATE_SQUELCH. Delay Squelch by 1 CLK480 cycle. 3: FUSE_SETUP_SEL. Select between regular CFG value and JTAG values for UX_SETUP 31 to 4: Reserved

26.5.2.53 USB1_UTMIP_XCVR_CFG1_0

UTMIP transceiver cell configuration register 1

UTMIP REGISTER: UTMIP_XCVR_CFG1

Offset: 838h | Read/Write: R/W | Reset: 0b00000000010000001000101010

Bit	Reset	Description
23:22	0x0	UTMIP_XCVR_SPARE: Spare bits for usb transceiver pad ECO.
21:18	0x0	UTMIP_XCVR_TERM_RANGE_ADJ: Range adjusment 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	0x1	UTMIP_FORCE_PDDR_POWERUP: Force PDDR input into power up.
4	0x0	UTMIP_FORCE_PDDR_POWERDOWN: Force PDDR input input into power down. (Overrides FORCE_PDDR_POWERUP.)
3	0x1	UTMIP_FORCE_PDCHRP_POWERUP: Force PDCHRP input into power up.
2	0x0	UTMIP_FORCE_PDCHRP_POWERDOWN: Force PDCHRP input input into power down. (Overrides FORCE_PDCHRP_POWERUP.)
1	0x1	UTMIP_FORCE_PDDISC_POWERUP: Force PDDISC input into power up.
0	0x0	UTMIP_FORCE_PDDISC_POWERDOWN: Force PDDISC input into power down. (Overrides FORCE_PDDISC_POWERUP.)

26.5.2.54 USB1_UTMIP_BIAS_CFG1_0

UTMIP Bias cell configuration register 1

UTMIP REGISTER: UTMIP_BIAS_CFG1

Offset: 83ch | Read/Write: R/W | Reset: 0b00000000101010

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.

Bit	Reset	Description
7:3	0x5	UTMIP_BIAS_PDTRK_COUNT: Control the BIAS cell power down lag. The lag should be 20us. For a Xtal clock of 13MHz it should be set a 5.
2	0x0	UTMIP_VBUS_WAKEUP_POWERDOWN: Force VBUS_WAKEUP input into power down.
1	0x1	UTMIP_FORCE_PDTRK_POWERUP: Force PDTRK input into power up.
0	0x0	UTMIP_FORCE_PDTRK_POWERDOWN: Force PDTRK input into power down. (Overrides FORCE_PDTRK_POWERUP.)

26.5.2.55 USB1_UTMIP_BIAS_STS0_0

UTMIP Bias cell status register 0

UTMIP REGISTER: UTMIP_BIAS_STS0

Offset: 840h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	UTMIP_TCTRL: Thermal encoding output from USB bias pad.
15:0	X	UTMIP_RCTRL: Thermal encoding output from USB bias pad.

26.5.3 USB2 Controller Registers

26.5.3.1 USB2_CONTROLLER_1_USB2D_ID_0

USB2D Identification Register

Offset: 000h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	REVISION: Revision number of the USB controller. This is set to 0x33.
15:8	X	NID: Ones complement version of ID. This field is set to 0xFA.
7:0	X	ID: Configuration number. This field is set to 0x05

26.5.3.2 USB2_CONTROLLER_1_USB2D_HW_HOST_0

USB2D Hardware Host Register

Offset: 008h | Read/Write: RO | Reset: 0bxxxx

Bit	Reset	Description
3:1	X	NPORT: VUSB_HS_NUM_PORT-1: This host controller has only 1 port. So this field will always be 0.
0	X	HC: VUSB_HS_HOST: Indicates support for host mode. Set to 1.

26.5.3.3 USB2_CONTROLLER_1_USB2D_HW_DEVICE_0

USB2D Hardware Device Register

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

Bit	Reset	Description

Bit	Reset	Description
5:1	X	DEVEP: VUSB_HS_DV_EP: No. of endpoints supported by this device controller. Set to 16. This includes control endpoint 0.
0	X	DC: Device capable: Set to 1 indicating support for device mode.

26.5.3.4 USB2_CONTROLLER_1_USB2D_HW_TXBUF_0

USB2D Hardware TX Buffer Register

Offset: 010h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	TXCHANADD: VUSB_HS_TX_CHAN_ADD: Total no. of address bits for the transmit buffer of each transmit endpoint. Set to 7. Each transmit buffer is 128 words deep.
15:8	X	TXADD: VUSB_HS_TX_ADD: Total no. of address bits for the transmit buffer. Set to 11. The total depth of the transmit buffer is 2048 words.
7:0	X	TCBURST: VUSB_HS_TX_BURST: Maximum burst size supported by the transmit endpoints for data transfers. Set to 8.

26.5.3.5 USB2_CONTROLLER_1_USB2D_HW_RXBUF_0

USB2D RX Buffer HW Parameters Register

Offset: 014h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:8	X	RXADD: VUSB_HS_RX_ADD: Total no. of address bits for the receive buffer. Set to 7. The total depth of the receive buffer is 128 words
7:0	X	RXBURST: VUSB_HS_RX_BURST: Maximum burst size supported by the receive endpoints for data transfers. Set to 8.

26.5.3.6 USB2_CONTROLLER_1_USB2D_CAPLENGTH_0

USB2D Capability Register Length Register

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

Bit	Reset	Description
7:0	X	CAPLENGTH: Indicates which offset to add to the register base address at the beginning of the Operational Register. Set to 0x40.

26.5.3.7 USB2_CONTROLLER_1_USB2D_HCIVERSON_0

USB2D Host Interface Version Number Register

Offset: 102h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	HCIVERSION: Contains a BCD encoding of the EHCI revision number supported by this host controller. The most significant byte of this register represents a major revision and the least significant byte is the minor revision. This host controller supports EHCI revision 1.00.

26.5.3.8 USB2_CONTROLLER_1_USB2D_HCSPARAMS_0

USB2D Host Control Structural Parameters Register

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

Bit	Reset	Description
27:24	X	N_TT: Number of Transaction Translators: indicates the number of embedded transaction translators associated with the USB2.0 host controller. This field is always set to 1 indicating only 1 embedded TT is implemented in this implementation. This is a non-EHCI field to support embedded TT.
23:20	X	N_PTT: Number of Ports per Transaction Translator: indicates the number of ports assigned to each transaction translator within the USB2.0 host controller. Field always equals N_PORTS. This is a non-EHCI field to support embedded TT.
15:12	X	N_CC: Number of Companion Controller: indicates the number of companion controllers. This field is set to 0.
11:8	X	N_PCC: Number of Ports per Companion Controller: indicates the number of ports supported per internal companion controller. This field is set to 0.
4	X	PPC: Port Power Control: indicates whether the host controller implementation includes port power control. 1 = Ports have port power switches 0= Ports do not have port power switches. This field affects the functionality of the port Power field in each port status and control register. This field is set to 1.
3:0	X	N_PORTS: Number of downstream ports. This field specifies the number of physical downstream ports implemented on this host controller. This field is fixed to 1, since this host controller only supports 1 port.

26.5.3.9 USB2_CONTROLLER_1_USB2D_HCCPARAMS_0

USB2D Host Control Capability Parameters Register

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

Bit	Reset	Description
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".

26.5.3.10 USB2_CONTROLLER_1_USB2D_DCIVERSION_0

USB2D Device Interface Version Number Register

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

Bit	Reset	Description
15:0	X	DCIVERSION: The device controller interface conforms to the two-byte BCD encoding of the interface version number contained in this register.

26.5.3.11 USB2_CONTROLLER_1_USB2D_DCCPARAMS_0

USB2D Device Control Capabilities Register

Offset: 124h | Read/Write: RO | Reset: 0bxxxxxxxxx

Bit	Reset	Description
8	X	HC: Host Capable: 1 = This controller is capable of operating as an EHCI compatible USB 2 0 host controller operating as an EHCI compatible USB 2.0 host controller. This field is set to 1.
7	X	DC: Device Capable: 1 = Controller is capable of operating as USB 2.0 device. This field is set to 1.
4:0	X	DEN: Device Endpoint Number: Number of endpoints built into the device controller. This is set to 16.

26.5.3.12 USB2_CONTROLLER_1_USB2D_USBCMD_0

USB2D USB Command Register

Offset: 140h | Read/Write: R/W | Reset: 0b00001000000x1x11x0000000

Bit	R/W	Reset	Description
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

Bit	R/W	Reset	Description
			1 = ENABLE
9:8	RW	0x3	ASP1_ASP0: Asynchronous Schedule Park Mode Count (OPTIONAL) Read/Write. If the Asynchronous Park Capability bit in the HCCPARAMS register is a one, then this field defaults to 3h and is R/W. Otherwise it defaults to zero and is RO. It contains a count of the number of successive transactions the host controller is allowed to execute from a high-speed queue head on the Asynchronous schedule before continuing traversal of the Asynchronous schedule. Valid values are 1h to 3h. Software must not write a zero to this bit when Park Mode Enable is a one as this will result in undefined behavior. This field is set to 3h in this implementation and is Read/Write capable.
7	RO	X	LR: Light Host/Device Controller Reset (OPTIONAL) . Read Only. Not Implemented. This field will always be "0".
6	RW	0x0	IAA: Interrupt on Async Advance Doorbell. When the host controller has evicted all appropriate cached schedule states, it sets the Interrupt on Async Advance status bit in the USBSTS register. If the Interrupt on Sync Advance Enable bit in the USBINTR register is one, then the host controller will assert an interrupt at the next interrupt threshold. The host controller sets this bit to zero after it has set the Interrupt on Sync Advance status bit in the USBSTS register to one. Software should not write a one to this bit when the asynchronous schedule is inactive. Doing so will yield undefined results. This bit is only used in host mode. Writing a one to this bit when device mode is selected will have undefined results. 0 = CLEAR 1 = SET
5	RW	0x0	ASE: Asynchronous Schedule Enable. This bit controls whether the host controller skips processing the Asynchronous Schedule. 0 = Do not process the Asynchronous Schedule. 1 = Use the ASYNCLISTADDR register to access the Asynchronous Schedule. Only the host controller uses this bit. 0 = DISABLE 1 = ENABLE
4	RW	0x0	PSE: Periodic Schedule Enable.This bit controls whether the host controller skips processing the Periodic Schedule. 0 = Do not process the Periodic Schedule 1 = Use the PERIODICLISTBASE register to access the Periodic Schedule. Only the host controller uses this bit. 0 = DISABLE 1 = ENABLE
3:2	RW	0x0	FS1_FS0: Frame List Size . (Read/Write). 000 = Default This field is Read/Write only if Programmable Frame List Flag in the HCCPARAMS registers is set to one. Hence this field is Read/Write for this implementation. This field specifies the size of the frame list that controls which bits in the Frame Index Register should be used for the Frame List Current index. Note that this field is made up from USBCMD bits 15, 3 and 2. 000 = 1024 elements (4096 bytes) Default value 001 = 512 elements (2048 bytes) 010 = 256 elements (1024 bytes) 011 = 128 elements (512 bytes) 100 = 64 elements (256 bytes) 101 = 32 elements (128 bytes) 110 = 16 elements (64 bytes) 111 = 8 elements (32 bytes) Only the host controller uses this field.
1	RW	0x0	RST: Controller Reset. Software uses this bit to reset the controller. This bit is set to zero by the Host/Device Controller when the reset process is complete. Software cannot terminate the reset process early by writing a zero to this register. Host Controller: When software writes a one to this bit, the Host Controller resets its internal pipelines, timers, counters, state machines etc. to their 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

Bit	R/W	Reset	Description
			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

26.5.3.13 USB2_CONTROLLER_1_USB2D_USBSTS_0

USB2D USB Status Register

Offset: 144h | Read/Write: R/W | Reset: 0b00xx00x1x0x0000x0000

Bit	R/W	Reset	Description
19	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
18	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
16	RO	X	NAKI: NAK Interrupt Bit Read Only. This bit is readonly. It is set by hardware when for a particular endpoint both the TX/RX Endpoint NAK bit and the corresponding TX/RX Endpoint NAK Enable bit are set. This bit is automatically cleared by hardware when the all the enabled TX/RX Endpoint NAK bits are cleared. 0 = DISABLE 1 = ENABLE
15	RW	0x0	AS: Asynchronous Schedule Status. This bit reports the current real status of the Asynchronous Schedule. When set to zero the asynchronous schedule status is disabled and if set to one the status is enabled. The Host Controller is not required to immediately disable or enable the Asynchronous Schedule when software transitions the Asynchronous Schedule Enable bit in the USBCMD register. If AS = ASE: 1= Enable Asynchronous Schedule 0= Disable Asynchronous Schedule Only used by the host controller. 0 = DISABLE 1 = ENABLE
14	RW	0x0	PS: Periodic Schedule Status. This bit reports the current real status of the Periodic Schedule. When set to zero the periodic schedule is disabled, and if set to one the status is enabled. The Host Controller is not required to immediately disable or enable the Periodic Schedule when software transitions the Periodic Schedule Enable bit in the USBCMD register. If PS = PSE then: 1 = Periodic Schedule is enabled or 0 = Periodic Schedule is disabled Only used by the host controller. 0 = DISABLE 1 = ENABLE
13	RO	X	RCL: Reclamation. This is a read-only status bit used to detect an empty asynchronous schedule. Only used by the host controller. 0 = DISABLE 1 = ENABLE
12	RW	0x1	HCH: HCHalted. 1 = Default. This bit is a zero whenever the Run/Stop bit is a one. The Host Controller sets this bit to one after it has stopped executing because of the Run/Stop bit being set to 0, either by software or by the Host Controller hardware (e.g. internal error). Only used by the host controller. 0 = UNHALTED

Bit	R/W	Reset	Description
			1 = HALTED
10	RW	0x0	ULPI_INT: ULPI Interrupt. This bit is set whenever an interrupt is received from ULPI PHY. Software writes 1 to clear it. 0 = NOT_ULPI_INT 1 = ULPI_INT
8	RW	0x0	SLI: DCSuspend. When a device controller enters a suspend state from an active state, this bit will be set to a 1. The device controller clears the bit upon exiting from a suspend state. Only used by the device controller. 0 = NOTSUSPEND 1 = SUSPENDED
7	RW	0x0	SRI: SOF Received. When the device controller detects a Start Of (micro) Frame, this bit will be set to a one. When a SOF is extremely late, the device controller will automatically set this bit to indicate that an SOF was expected. Therefore, this bit will be set roughly every 1ms in device FS mode and every 125us in HS mode and will be synchronized to the actual SOF that is received. Since device controller is initialized to FS before connect, this bit Will be set at an interval of 1ms during the prelude to the connect and chirp. In host mode, this bit will be set every 125us and can be used by host controller driver as a time base. Software writes a 1 to this bit to clear it. This is a non-EHCI status bit. 0 = SOF_NOT_RCVD 1 = SOF_RCVD
6	RW	0x0	URI: USB Reset Received. When the device controller detects a USB Reset and enters the default state, this bit is set to a 1. Software can write a 1 to this bit to clear the USB Reset Received status bit. Only used by the device controller. 0 = NO_USB_RESET 1 = USB_RESET
5	RW	0x0	AAI: Interrupt and Asynchronous Advance. System software can force the host controller to issue an interrupt the next time the host controller advances the asynchronous schedule by writing a one to the Interrupt on Async Advance Doorbell bit in the USBCMD register. This status bit indicates the assertion of that interrupt source. Only used by the host controller 0 = NOT_ADVANCED 1 = ADVANCED
4	RO	X	SEI: System Error. This bit is not used in this implementation and will always be set to "0". 0 = NO_ERROR 1 = ERROR
3	RW	0x0	FRI: Frame List Rollover. The Host Controller sets this bit to a 1 when the Frame List Index rolls over from its maximum value to 0. The exact value at which the rollover occurs depends on the frame list size. For example. If the frame list size (as programmed in the Frame List Size field of the USBCMD register) is 1024, the Frame Index Register rolls over every time FRINDEX [1 3] toggles. Similarly, if the size is 512, the Host Controller sets this bit to a 1 every time FHINDEX [12] toggles. Only used by the host controller. 0 = NO_ROLLOVER 1 = ROLLOVER
2	RW	0x0	PCI: Port Change Detect. The Host Controller sets this bit to a 1 when on any port a Connect Status occurs, a Port Enable/Disable Change occurs, or the Force Port Resume bit is set as the result of a J-K transition on the suspended port. The Device Controller sets this bit to a one when the port controller enters the full or high-speed operational state. When the port controller exits the full or high-speed operational states due to Reset or Suspend events, the notification mechanisms are the USB Reset Received bit and the DCSuspend bits respectively. This bit is not EHCI compatible. 0 = NO_PORT_CHANGE 1 = PORT_CHANGE
1	RW	0x0	UEI: USB Error Interrupt. This bit gets set by the Host/Device controller when completion of a USB transaction results in an error condition. This bit is set along with the USBINT bit, if the TD on which the error interrupt occurred also ad its interrupt on complete (IOC) bit set. 0 = NO_ERROR 1 = ERROR
0	RW	0x0	UI: USB Interrupt. This bit is set by the Host/Device Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) as an interrupt on complete (IOC) bit set. This bit is also set by the Host/Device Controller when a short packet is detected. A

Bit	R/W	Reset	Description
			short packet is when the actual number of bytes received was less than the expected number of bytes. 0 = NO_INT 1 = INT

26.5.3.14 USB2_CONTROLLER_1_USB2D_USBINTR_0

USB2D USB Interrupt Enable Register

Offset: 148h | Read/Write: R/W | Reset: 0b00x0xxxxx0x000000000

Bit	Reset	Description
19	0x0	UPIE: UPIE Interrupt Enable. 1 = USB controller issues an interrupt if UPA bit in USBSTS register transitions. 0 = DISABLE 1 = ENABLE
18	0x0	UAIE: UAIE Interrupt Enable. 1 = USB controller issues an interrupt if UAI bit in USBSTS register transitions. 0 = DISABLE 1 = ENABLE
16	0x0	NAKE: NAK Interrupt Enable. 1 = USB controller issues an interrupt if NAKI bit in USBSTS register transitions. 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 SW by writing a 1 to the ULPI_INT bit. 0 = DISABLE 1 = ENABLE
8	0x0	SLE: Sleep Enable. 1 = Device controller issues an interrupt if DCSuspend bit in USBSTS register transitions. The interrupt is acknowledged by SW by writing a 1 to the DCSuspend bit. Only used by the device controller. 0 = DISABLE 1 = ENABLE
7	0x0	SRE: SOF Received Enable. 1 = Device controller issues an interrupt if SOF Received bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the SOF Received bit. 0 = DISABLE 1 = ENABLE
6	0x0	URE: USB Reset Enable.1 = Device controller issues an interrupt if USB Reset Received bit in USBSTS register = 1 The interrupt is acknowledged by software clearing the USB Reset Received bit. Only used by the device controller. 0 = DISABLE 1 = ENABLE
5	0x0	AAE: Interrupt on Asynchronous Advance Enable. 1 = the host controller issues an interrupt at the next interrupt threshold if Interrupt on Async Advance bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the Interrupt on Async Advance bit. Only used by the host controller. 0 = DISABLE 1 = ENABLE
4	0x0	SEE: System Error Enable. 1 = Host/device controller issues an interrupt if the System Error bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the System Error bit. 0 = DISABLE 1 = ENABLE
3	0x0	FRE: Frame List Rollover Enable. 1 = Host controller issues an interrupt if Frame List Rollover bit in the USBSTS register = 1. The interrupt is acknowledged by software clearing the Frame List Rollover bit. Only used by the host controller. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	0x0	PCE: Port Change Detect Enable. 1 = Host/device controller issues an interrupt if Port Change Detect bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the Port Change Detect bit. 0 = DISABLE 1 = ENABLE
1	0x0	UEE: USB Error Interrupt Enable. 1 = Host controller issues an interrupt at the next interrupt threshold if the USBERRINT bit in USBSTS = 1. The interrupt is acknowledged by software clearing the USBERRINT bit in the USBSTS register. 0 = DISABLE 1 = ENABLE
0	0x0	UE: USB Interrupt Enable. 1 = Host/device issues an interrupt at the next interrupt threshold if the USBINT bit in USBSTS = 1. The interrupt is acknowledged by software clearing the USBINT bit. 0 = DISABLE 1 = ENABLE

26.5.3.15 USB2_CONTROLLER_1_USB2D_FRINDEX_0

USB2D USB Frame Index Register

Offset: 14ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxx

Bit	Reset	Description
13:0	X	FRINDEX: Frame Index. The value in this register increments at the end of each time frame (micro-frame). Bits [N: 3] are used for the Frame List current index. Each location of the frame list is accessed 8 times (frames or micro-frames) before moving to the next index. The following illustrates values of N based on the value of the Frame List Size field in the USBCMD register, when used in host mode. USBCMD [Frame List Size] Number Elements N 000b (1024) 12 001b (512) 11 010b (256) 10 011b (128) 9 100b (64) 8 101b (32) 7 110b (16) 6 111b (8) 5 In device mode the value is the current frame number of the last frame transmitted. It is not used as an index. In either mode bits 2:0 indicate the current micro-frame.

26.5.3.16 USB2_CONTROLLER_1_USB2D_PERIODICLISTBASE_0

USB2D Host Controller Frame List Base Address Register

Offset: 154h | Read/Write: R/W | Reset: 0b00000000000000000000

Bit	Reset	Description
31:25	0x0	USBADR: Device mode. The upper seven bits of this register represent the device address. After any controller reset or a USB reset, the device address is set to the default address (0). The default address will match all incoming addresses. Software shall reprogram the address after receiving a SET_ADDRESS request.
24	0x0	USBADRA: Device Address Advance. Default=0. When this bit is 0, any writes to USBADR are instantaneous. When this bit is written to a 1 at the same time or before USBADR is written, the write to the USBADR field is staged and held in a hidden register. After an IN occurs on endpoint 0 and is ACKed, USBADR will be loaded from the holding register. Hardware will automatically clear this bit on the following conditions: 1) IN is ACKed to endpoint 0. (USBADR is updated from staging register). 2) OUT/SETUP occur to endpoint 0. (USBADR is not updated). 3) Device Reset occurs (USBADR is reset to 0). Note: After the status phase of the SET_ADDRESS descriptor, the DCD has 2 ms to program the USBADR field. This mechanism will ensure this specification is met when the DCD can not write of the device address within 2ms from the SET_ADDRESS status phase. If the DCD writes the USBADR with USBADRA=1 after the SET_ADDRESS data phase (before the prime of the status phase), the USBADR will be programmed instantly at the correct time and meet the 2ms USB requirement.
31:12	0x0	BASEADR: Host mode: This 32-bit register contains the beginning address of the Periodic Frame List in the system memory. HCD loads this register prior to starting the schedule execution by the Host Controller. The memory structure referenced by this physical memory pointer is assumed to be 4-Kbyte aligned. The contents of this register are combined with the Frame Index Register (FRINDEX) to enable the Host Controller to step through the Periodic Frame List in sequence. Base Address (Low). These bits correspond to memory address signals [31:12], respectively. Only used by the host controller.

26.5.3.17 USB2_CONTROLLER_1_USB2D_ASYNCLISTADDR_0

USB2D Next Asynchronous List Address Register

Offset: 158h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:11	0x0	EPBASE: Device mode. This register contains the address of the top of the endpoint list in system memory. These bits correspond to memory address signals [31:11], respectively. This field will reference a list of up to 32 Queue Heads (QH). Only used by the device controller.
31:5	0x0	ASYBASE: Host mode. This 32-bit register contains the address of the next asynchronous queue head to be executed by the host. Link Pointer Low (LPL). These bits correspond to memory address signals [31:5], respectively. This field may only reference a Queue Head (OH). Only used by the host controller.

26.5.3.18 USB2_CONTROLLER_1_USB2D_ASYNCTTSTS_0

USB2D Asynchronous Buffer Status for Embedded TT Register

Offset: 15ch | Read/Write: R/W | Reset: 0b0x

Bit	R/W	Reset	Description
1	RW	0x0	TTAC: Embedded TT Async Buffers Clear. (Read/Write to set) This field will clear all pending transactions in the embedded TT Async Buffer(s). The clear will take as much time as necessary to clear buffer without interfering with a transaction in progress. TTAC will return to zero after being set by software only after the actual clear occurs.
0	RO	X	TTAS: Embedded TT Async Buffers Status. (Read Only) This read only bit will be 1 if one or more transactions are being held in the embedded TT Async. Buffers. When this bit is a zero, then all outstanding transactions in the embedded TT have been flushed.

26.5.3.19 USB2_CONTROLLER_1_USB2D_BURSTSIZE_0

USB2D Burst Size register

Offset: 160h | Read/Write: R/W | Reset: 0b0000100000001000

Bit	Reset	Description
15:8	0x8	TXPBURST: Programmable TX Burst Length. (Read/Write) This register represents the maximum length of a burst in 32-bit words while moving data from system memory to the USB bus.
7:0	0x8	RXPBURST: Programmable RX Burst Length. (Read/Write) This register represents the maximum length of a burst in 32-bit words while moving data from the USB bus to system memory.

26.5.3.20 USB2_CONTROLLER_1_USB2D_TXFILLTUNING_0

USB2D Transmit fill tuning register

Offset: 164h | Read/Write: R/W | Reset: 0b000010xxx0000000000000

Bit	Reset	Description
21:16	0x2	TXFIFOTHRES: FIFO Burst Threshold. (Read/Write) This register controls the number of data bursts that are posted to the TX latency FIFO in host mode before the packet begins on to the bus. The minimum value is 2 and this value should be a low as possible to maximize USB performance. A higher value can be used in systems with unpredicable latency and/or insufficient bandwidth where the FIFO may underrun because the data transferred from the latency FIFO to USB occurs before it can be replenished from system memory. This value is ignored if the Stream Disable bit in USBMODE register is set.
12:8	0x0	TXSCHHEALTH: Scheduler Health Counter. (Read/Write To Clear) [Default = 0] This register increments when the host controller fails to fill the TX latency FIFO to the level programmed by

Bit	Reset	Description
		TXFIFOTHRES before running out of time to send the packet before the next Start-Of-Frame. This health counter measures the number of times this occurs to provide feedback to selecting a proper TXSCHOH. Writing to this register will clear the counter and this counter will max. at 31.
7:0	0x0	TXSCHOH: Scheduler Overhead. (Read/Write) [Default = 0] This register adds an additional fixed offset to the schedule time estimator described above as Tff. As an approximation, the value chosen for this register should limit the number of back-off events captured in the TXSCHHEALTH to less than 10 per second in a highly utilized bus. Choosing a value that is too high for this register is not desired as it can needlessly reduce USB utilization. The time unit represented in this register is 1.267us when a device is connected in High-Speed Mode. The time unit represented in this register is 6.333us when a device is connected in Low/Full Speed Mode

26.5.3.21 USB2_CONTROLLER_1_USB2D_ICUSB_CTRL_0

USB2D ICUSB control register. This register enables and controls the ICUSB FS/LS transceiver.

Offset: 16ch | Read/Write: R/W | Reset: 0b0000

Bit	Reset	Description
3	0x0	IC_ENB1: ICUSB transceiver enable. This bit enables the ICUSB transceiver . To enable the interface, the bits PTS must be set to 11 in the PORTSCx. Writing a '1' to this bit selects the IC_USB interface. 0 = DISABLE 1 = ENABLE
2:0	0x0	IC_VDD1: ICUSB voltage select. It selects which voltage is being supplied to the ICUSB peripheral. 000 -> No voltage 001 -> 1.0V - reserved 010 -> 1.2V - reserved 011 -> 1.5V - reserved 100 -> 1.8V 101 -> 3.0V 110 -> reserved 111 -> reserved The Voltage negotiation should happen between enabling port power (PP) and asserting the run/stop bit in register.

26.5.3.22 USB2_CONTROLLER_1_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.

CAUTION: 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 theoperation 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 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 Control register.

Please refer to the ULPI Specification Revision 1.1 for further information on the use of the optional features.

Offset: 170h | Read/Write: R/W | Reset: 0b000xx00000000000xxxxxxxx00000000

Bit	R/W	Reset	Description
31	RW	0x0	ULPI_WAKEUP: ULPI Wakeup. Writing the 1 to this bit will begin the wakeup operation. The bit will automatically transition to 0 after the wakeup is complete. Once this bit is set, the driver cannot set it back to 0. Note: The driver must never execute a wakeup and a read/write operation at the same time. 0 = CLEAR 1 = SET
30	RW	0x0	ULPI_RUN: ULPI read/write Run. Writing the 1 to this bit will begin the read/write operation. The bit will automatically transition to 0 after the read/write is complete. Once this bit is set, the driver cannot set it back to 0. Note: The driver must never execute a wakeup and a read/write operation at the same time. 0 = CLEAR 1 = SET
29	RW	0x0	ULPI_RD_WR: ULPI read/write control. (0) Read; (1) Write. This bit selects between running a read or write operation. 0 = READ 1 = WRITE
27	RO	X	ULPI_SYNC_STATE: ULPI sync state. (1) Normal Sync. State. (0) In another state (i.e. carkit, serial, low power) This bit represents the state of the ULPI interface. 0 = NOT_NORMAL 1 = NORMAL
26:24	RW	0x0	ULPI_PORT: ULPI PHY port no. This field should be always written as 0.
23:16	RW	0x0	ULPI_REG_ADDR: ULPI PHY register address. When doing a read or write operation to the ULPI PHY, the address of the ULPI PHY register being accessed is written to this field.
15:8	RO	X	ULPI_DATA_RD: ULPI PHY data read. The data from the ULPI PHY register can be read from here after the read operation completes.
7:0	RW	0x0	ULPI_DATA_WR: ULPI PHY data write. The data to write to the ULPI PHY register is written here.

26.5.3.23 USB2_CONTROLLER_1_USB2D_ENDPTNAK_0

USB2D Endpoint NAK register

Offset: 178h | 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

26.5.3.24 USB2_CONTROLLER_1_USB2D_ENDPTNAK_ENABLE_0

USB2D Endpoint NAK Enable register

Offset: 17ch | 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

26.5.3.25 USB2_CONTROLLER_1_USB2D_PORTSC1_0

USB2D Port Status/Control 1 Register

Offset: 184h | Read/Write: R/W | Reset: 0b000xxx0000000000xxx1xxx0x0xx010x

Bit	R/W	Reset	Description
31:30	RW	0x0	PTS: Parallel transceiver select. This bit is not defined in the EHCI specification. 0 = UTMI 1 = RESERVED 2 = ULPI 3 = ICUSB_SER
29	RW	0x0	STS: 0 = 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
28	RO	X	PTW: Parallel Transceiver Width. Fixed to 0. This bit is not defined in the EHCI specification. 0 = EIGHT_BIT 1 = RESERVED
27:26	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

Bit	R/W	Reset	Description
25	RW	0x0	SRT: Shorten USB Reset Time. Software should never set this to 1.
24	RW	0x0	PFSC: Port Force Full Speed Connect: Writing this bit to a 1b forces the port to connect at Full Speed only. It disables the chirp sequence that allows the port to identify itself as High Speed. This is useful for testing FS configurations with a HS host, hub or device. This bit is not defined in the EHCI specification. 0 = DONT_FORCE_FULL_SPEED 1 = FORCE_FULL_SPEED
23	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. In device mode, the PHY can be put into Low Power Suspend - Clock disable when the device is not running (USBCMD Run/Stop = 0) or the host has signaled suspend (PORTSC SUSPEND = 1). Low power suspend will be cleared automatically when the host has signaled resume. Before forcing a resume from the device, the device 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. This bit is not defined in the EHCI specification. 0 = DISBLE 1 = ENABLE
22	RW	0x0	WKOC: Default = 0b. Wake on Over-current Enable: Writing this bit to a one enables the port to be sensitive to over-current conditions as wake-up events. This field is zero if Port Power(PP) is zero. This bit should only be used when operating in Host mode. Writing this bit to 1 while the controller is working in device mode can result in undefined behavior. 0 = DISBLE 1 = ENABLE
21	RW	0x0	WKDS: Wake on Disconnect Enable: Writing this bit to a one enables the port to be sensitive to device disconnects as wake-up events. This field is zero if Port Power(PP) is zero or in device mode. This bit should only be used when operating in Host mode. Writing this bit to 1 while the controller is working in device mode can result in undefined 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 = DISBLE 1 = ENABLE
20	RW	0x0	WKCN: Wake on Connect Enable: Writing this bit to a one enables the port to be sensitive to device connects as wake-up events. This field is zero if Port Power(PP) is zero or in device mode. This bit should only be used when operating in Host mode. Writing this bit to 1 while the controller is working in device mode can result in undefined 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 = DISBLE 1 = ENABLE
19:16	RW	0x0	PTC: Port Test Control: Any other value than zero indicates that the port is operating in test mode. Value Specific Test 0000b Not enabled 0001b J_ STATE 0010b K_STATE 0011b SEQ_NAK 0100b Packet 0101b FORCE_ENABLE 0110b to 1111b Reserved Refer to Chapter 7 of the USB Specification Revision 2.0 for details on each test mode. 0 = NORMAL_OP 1 = TEST_J 2 = TEST_K 3 = TEST_SE0_NAK 4 = TEST_PKT 5 = TEST_FORCE_ENABLE
15:14	RO	X	PIC: Port Indicator Control: This field is not supported in the current implementation. Please use a GPIO if you wish to use Port Indicators.
13	RO	X	PO: Port Owner. Port owner handoff is not implemented in this design, therefore this bit will 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 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

Bit	R/W	Reset	Description
			port is non-functional and will not report attaches, detaches, etc. When an over-current condition is detected on a powered port and PPC is a one, the PP bit in each affected port may be transitioned by the host controller driver from a one to a zero (removing power from the port). 0 = NOT_POWERED 1 = POWERED
11:10	RO	X	LS: Line state. These bits reflect the current logical levels of the D+ (bit 10) and D- (bit 11) signal lines. The encoding of the bits are: 00b = SE0 01b = J-state 10b = K-state 11b = Undefined The value of this field is undefined if Port Power(PP) is zero in host mode. In host mode, the use of line-state by the host controller driver is not necessary (unlike EHCI), because the port controller state machine and the port routing manage the connection of LS and FS. In device mode, the use of line-state by the device controller driver is not necessary. 0 = SE0 1 = J_STATE 2 = K_STATE 3 = UNDEFINED
9	RO	X	HSP: When the bit is one, the host/device connected to the port is in high-speed mode and if set to zero, the host/device connected to the port is not in a high-speed mode. Note: HSP is redundant with PSPD(27:26). This bit is not defined in the EHCI specification. 0 = NOT_HIGH_SPEED 1 = HIGH_SPEED
8	RW	0x0	PR: This field is zero if Port Power(PP) is zero. In Host Mode: Read/Write. 1=Port is in Reset. 0=Port is not in Reset. When software writes a one to this bit the bus-reset sequence as defined in the USB Specification Revision 2.0 is started. This bit will automatically change to zero after the reset sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the reset duration is timed in the driver. In Device Mode: This bit is a read only status bit. Device reset from the USB bus is also indicated in the USBSTS register. 0 = NOT_USB_RESET 1 = USB_RESET
7	RO	X	SUSP: Port suspend. 1=Port in suspend state. 0=Port not in suspend state. In Host Mode: Read/Write. Port Enabled Bit and Suspend bit of this register define the port states as follows: Bits [Port Enabled, Suspend] Port State 0x Disable 10 Enable 11 Suspend When in suspend state, downstream propagation of data is blocked on this port, except for port reset. The blocking occurs at the end of the current transaction if a transaction was in progress when this bit was written to 1. In the suspend state, the port is sensitive to resume detection. Note that the bit status does not change until the port is suspended and that there may be a delay in suspending a port if there is a transaction currently in progress on the USB. The host controller will unconditionally set this bit to zero when software sets the Force Port Resume bit to zero. A write of zero to this bit is ignored by the host controller. If host software sets this bit to a one when the port is not enabled (i.e. Port enabled bit is a zero) the results are undefined. This field is zero if Port Power(PP) is zero in host mode. In Device Mode: Read Only. This bit is a read only status bit. 0 = NOT_SUSPEND 1 = SUSPEND
6	RW	0x0	FPR: Force Port Resume. 1= Resume detected/driven on port. 0=No resume (K state) detected/driven on port. In Host Mode: Software sets this bit to one to drive resume signaling. The Host Controller sets this bit to one if a J-to-K transition is detected while the port is in the Suspend state. When this bit transitions to a one because a J-to-K transition is detected, the Port Change Detect bit in the USBSTS register is also set to one. This bit will automatically change to zero after the resume sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the resume duration is timed in the driver. Note that when the Host controller owns the port, the resume sequence follows the defined sequence documented in the USB Specification Revision 2.0. The resume signaling (Full-speed 'K') is driven on the port as long as this bit remains a one. This bit remains a one until the port has switched to the high-speed idle. Writing a zero has no effect because the port controller will time 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.

Bit	R/W	Reset	Description
			0 = NO_RESUME 1 = RESUME
5	RO	X	OCC: Over-current Change: Not supported 0 = NO_CHANGE 1 = CHANGE
4	RO	X	OCA: Over-current Active: Not supported 0 = NO_OVER_CURRENT 1 = OVER_CURRENT
3	RW	0x0	PEC: Port Enable/Disable Change: 1=Port enabled/disabled status has changed. 0=No change. In Host Mode: For the root hub, this bit gets set to a one only when a port is disabled due to disconnect on the port or due to the appropriate conditions existing at the EOF2 point (See Chapter 11 of the USB Specification). Software clears this by writing a one to it. This field is zero if Port Power(PP) is zero. In Device mode: The device port is always enabled. (This bit will be zero) 0 = NO_CHANGE 1 = CHANGE
2	RW	0x1	PE: Port Enabled/Disabled: 1=Enable. 0=Disable (default) In Host Mode: Ports can only be enabled by the host controller as a part of the reset and enable. Software cannot enable a port by writing a one to this field. Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by the host software. Note that the bit status does not change until the port state actually changes. There may be a delay in disabling or enabling a port due to other host controller and bus events. When the port is disabled, (0b) downstream propagation of data is blocked except for reset. This field is zero if Port Power(PP) is zero in host mode. In Device Mode: The device port is always enabled. (This bit will be one) 0 = PORT_DISABLED 1 = PORT_ENABLED
1	RW	0x0	CSC: Connect Status Change: 1 =Change in Current Connect Status. 0=No change (default) In Host Mode: Indicates a change has occurred in the port's Current Connect Status. The host/device controller sets this bit for all changes to the port device connect status, even if system software has not cleared an existing connect status change. For example, the insertion status changes twice before system software has cleared the changed condition, hub hardware will be 'setting' an already-set bit (i.e., the bit will remain set). Software clears this bit by writing a one to it. This field is zero if Port Power(PP) is zero in host mode. This bit is undefined in device controller mode. 0 = NO_CHANGE 1 = CHANGE
0	RO	X	CCS: Current Connect Status: In Host Mode: 1=Device is present on port. 0=No device is present (default) This value reflects the current state of the port, and may not correspond directly to the event that caused the Connect Status Change bit (Bit 1) to be set. This field is zero if Port Power(PP) is zero in host mode. In Device Mode: 1=Attached 0=Not Attached (default) A one indicates that the device successfully attached and is operating in either high speed or full speed as indicated by the High Speed Port bit in this register. A zero indicates that the device did not attach successfully or was forcibly disconnected by the software writing a zero to the Run bit in the USBCMD register. It does not state the device being disconnected or suspended. 0 = NOT_CONNECTED 1 = CONNECTED

26.5.3.26 USB2_CONTROLLER_1_USB2D_OTGSC_0

USB2D Status and Control Register

Offset: 1a4h | Read/Write: R/W | Reset: 0b0000000x0000000xxxxxxxxxx000x00

Bit	R/W	Reset	Description
30	RW	0x0	DPIE: Data Pulse Interrupt Enable. Setting this bit enables the Data pulse interrupt. 0 = DISABLE 1 = ENABLE
29	RW	0x0	ONEMSE: 1 millisecond timer Interrupt enable. Setting this bit enables the 1 millisecond timer interrupt.

Bit	R/W	Reset	Description
			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
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

Bit	R/W	Reset	Description
			0 = STS_CLEAR 1 = STS_SET
11	RO	X	BSV: B Session Valid. Indicates VBus is above the B session valid threshold 0 = STS_CLEAR 1 = STS_SET
10	RO	X	ASV: A Session Valid. Indicates VBus is above the A session valid threshold 0 = STS_CLEAR 1 = STS_SET
9	RO	X	AVV: A VBus Valid. Indicates VBus is above the A VBus valid threshold 0 = STS_CLEAR 1 = STS_SET
8	RO	X	ID: USB ID: 0 = A-device 1 = B-device 0 = A_DEV 1 = B_DEV
5	RW	0x0	IDPU: USB ID Pullup 0 = CLEAR 1 = SET
4	RW	0x0	DP: Data Pulsing. Setting this bit causes the pull-up on DP to be asserted for data pulsing during SRP. 0 = NO_DATA_PULSE 1 = DATA_PULSE
3	RW	0x0	OT: Termination. This bit must be set when the 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

26.5.3.27 USB2_CONTROLLER_1_USB2D_USBMODE_0

USB2D USB Device Mode Register

Offset: 1a8h | Read/Write: RO | Reset: 0bxxxxx

Bit	Reset	Description
4	X	SDIS: Stream disable: 1 Streaming is disabled - helpful to avoid overrun/underruns when system load is too high. 0 = STREAM_ENABLE 1 = STREAM_DISABLE
3	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_OFF 1 = LOCKOUT_ON
2	X	ES: Endian Select: Note: For this implementation, this should be always set to 0 (little endian). 0 = LITTLE_ENDIAN 1 = RESERVED
1:0	X	CM: Controller Mode: The controller mode will default to an idle state and will need to be initialized to the

Bit	Reset	Description
		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

26.5.3.28 USB2_CONTROLLER_1_USB2D_ENDPTSETUPSTAT_0

USB2D Endpoint Setup Status Register

Offset: 1ach | Read/Write: R/W | Reset: 0b0000000000000000

Bit	Reset	Description
15	0x0	ENDPTSETUPSTAT15: Endpoint 15 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
14	0x0	ENDPTSETUPSTAT14: Endpoint 14 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
13	0x0	ENDPTSETUPSTAT13: Endpoint 13 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
12	0x0	ENDPTSETUPSTAT12: Endpoint 12 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
11	0x0	ENDPTSETUPSTAT11: Endpoint 11 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
10	0x0	ENDPTSETUPSTAT10: Endpoint 10 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
9	0x0	ENDPTSETUPSTAT9: Endpoint 9 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup

Bit	Reset	Description
		data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
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
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

Bit	Reset	Description
		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

26.5.3.29 USB2_CONTROLLER_1_USB2D_ENDPTPRIME_0

USB2D Endpoint Initialization Register

Offset: 1b0h | 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 endpiont. 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 endpiont. 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 endpiont. 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 endpiont. 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 endpiont. 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 endpiont. 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
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 endpiont. 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 endpiont. 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 endpiont. 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 endpiont. 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 endpiont. 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 endpiont. 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 endpiont. 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 endpiont. 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 endpiont. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer.

Bit	Reset	Description
		Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
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 endpiont. 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
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

Bit	Reset	Description
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
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.

Bit	Reset	Description
		Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME

26.5.3.30 USB2_CONTROLLER_1_USB2D_ENDPTFLUSH_0

USB2D Endpoint De-Initialization Register

Offset: 1b4h | 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
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

Bit	Reset	Description
		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
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

Bit	Reset	Description
		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
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

Bit	Reset	Description
		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

26.5.3.31 USB2_CONTROLLER_1_USB2D_ENDPTSTATUS_0

USB2D Endpoint Status Register

Offset: 1b8h | 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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
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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by

Bit	Reset	Description
		the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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

Bit	Reset	Description
		endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode.

Bit	Reset	Description
		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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY

26.5.3.32 USB2_CONTROLLER_1_USB2D_ENDPTCOMPLETE_0

USB2D Endpoint Complete Register

Offset: 1bch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ETCE15: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
30	0x0	ETCE14: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
29	0x0	ETCE13: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
28	0x0	ETCE12: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
27	0x0	ETCE11: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
26	0x0	ETCE10: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) 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

Bit	Reset	Description
		1 = COMPLETE
22	0x0	ETCE6: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
21	0x0	ETCE5: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
20	0x0	ETCE4: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
19	0x0	ETCE3: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
18	0x0	ETCE2: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
17	0x0	ETCE1: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
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

Bit	Reset	Description
		USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
12	0x0	ERCE12: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
11	0x0	ERCE11: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
10	0x0	ERCE10: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
9	0x0	ERCE9: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
8	0x0	ERCE8: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
7	0x0	ERCE7: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 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

Bit	Reset	Description
		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

26.5.3.33 USB2_CONTROLLER_1_USB2D_ENDPTCTRL0_0

USB2D Endpoint Control 0 Register

Offset: 1c0h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23	X	TXE: TX Endpoint Enable. Endpoint 0 is always enabled. 0 = DISABLE 1 = ENABLE
19:18	X	TXT: TX Endpoint Type. Endpoint0 is fixed as a Control Endpoint. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
16	X	TXS: TX Endpoint Stall: Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request. 0 = EP_OK 1 = EP_STALL
7	X	RXE: RX Endpoint Enable. Endpoint 0 is always enabled. 0 = DISABLE 1 = ENABLE
3:2	X	RXT: RX Endpoint Type. Endpoint 0 is fixed as a Control Endpoint. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
0	X	RXS: RX Endpoint Stall: Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request. 0 = EP_OK 1 = EP_STALL

26.5.3.34 USB2_CONTROLLER_1_USB2D_ENDPTCTRL1_0

USB2D Endpoint Control 1 Register

Offset: 1c4h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above, 0 = EP_OK

Bit	R/W	Reset	Description
			1 = EP_STALL

26.5.3.35 USB2_CONTROLLER_1_USB2D_ENDPTCTRL2_0

USB2D Endpoint Control 2 Register

Offset: 1c8h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.

Bit	R/W	Reset	Description
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

26.5.3.36 USB2_CONTROLLER_1_USB2D_ENDPTCTRL3_0

USB2D Endpoint Control 3 Register

Offset: 1cch | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL

Bit	R/W	Reset	Description
			1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

26.5.3.37 USB2_CONTROLLER_1_USB2D_ENDPTCTRL4_0

USB2D Endpoint Control 4 Register

Offset: 1d0h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.

Bit	R/W	Reset	Description
			0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

26.5.3.38 USB2_CONTROLLER_1_USB2D_ENDPTCTRL5_0

USB2D Endpoint Control 5 Register

Offset: 1d4h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING

Bit	R/W	Reset	Description
			1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

26.5.3.39 USB2_CONTROLLER_1_USB2D_ENDPTCTRL6_0

USB2D Endpoint Control 6 Register

Offset: 1d8h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE

Bit	R/W	Reset	Description
			1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

26.5.3.40 USB2_CONTROLLER_1_USB2D_ENDPTCTRL7_0

USB2D Endpoint Control 7 Register

Offset: 1dch | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above.

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 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

26.5.3.41 USB2_CONTROLLER_1_USB2D_ENDPTCTRL8_0

USB2D Endpoint Control 8 Register

Offset: 1e0h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.

Bit	R/W	Reset	Description
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This 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

26.5.3.42 USB2_CONTROLLER_1_USB2D_ENDPTCTRL9_0

USB2D Endpoint Control 9 Register

Offset: 1e4h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL

Bit	R/W	Reset	Description
			1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

26.5.3.43 USB2_CONTROLLER_1_USB2D_ENDPTCTRL10_0

USB2D Endpoint Control 10 Register

Offset: 1e8h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.

Bit	R/W	Reset	Description
			0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

26.5.3.44 USB2_CONTROLLER_1_USB2D_ENDPTCTRL11_0

USB2D Endpoint Control 11 Register

Offset: 1ech | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING

Bit	R/W	Reset	Description
			1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

26.5.3.45 USB2_CONTROLLER_1_USB2D_ENDPTCTRL12_0

USB2D Endpoint Control 12 Register

Offset: 1f0h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK

Bit	R/W	Reset	Description
			1 = EP_STALL

26.5.3.46 USB2_CONTROLLER_1_USB2D_ENDPTCTRL13_0

USB2D Endpoint Control 13 Register

Offset: 1f4h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.

Bit	R/W	Reset	Description
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

26.5.3.47 USB2_CONTROLLER_1_USB2D_ENDPTCTRL14_0

USB2D Endpoint Control 14 Register

Offset: 1f8h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL

Bit	R/W	Reset	Description
			1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

26.5.3.48 USB2_CONTROLLER_1_USB2D_ENDPTCTRL15_0

USB2D Endpoint Control 15 Register

Offset: 1fch | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.

Bit	R/W	Reset	Description
			0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

26.5.4 USB2 Controller Interface Registers

Interface registers for USB2 controller. These are used to generate the actual RTL registers for USB2 controller interface.

26.5.4.1 USB2_IF_USB_SUSP_CTRL_0

USB suspend control register. This register controls the suspend and resume behavior of USB controller/PHY.

Offset: 400h | Read/Write: R/W | Reset: 0b000xx000000xx000000

Bit	R/W	Reset	Description
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.
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	UHSIC_PHY_ENB: Enable UHSIC PHY mode. Set this to 1 if using UHSIC PHY. Otherwise set this to 0. 0 = DISABLE 1 = ENABLE
11	RW	0x0	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
10	RW	0x0	UHSIC_SUSP_POL: Polarity of the suspend signal going to UHSIC PHY. 0 = Active low (default) 1 = Active high This should not be changed by software. 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
9	RW	0x0	USB_PHY_CLK_VALID_INT_ENB: USB PHY clock valid interrupt enable If this bit is enabled, interrupt is generated whenever USB clocks are resumed from a suspend. 0 = DISABLE 1 = ENABLE
8	RO	0x0	USB_PHY_CLK_VALID_INT_STS: USB PHY clock valid interrupt status This bit is set whenever USB PHY clock is waked up from suspend. Software must write a 1 to clear this bit. 0 = UNSET

Bit	R/W	Reset	Description
			1 = SET
7	RO	X	USB_PHY_CLK_VALID: USB PHY clock valid status This bit indicates whether the USB PHY is generating a valid clock to the USB controller. If USB PHY clock is running, this bit is set to 1, else it is set to 0. 0 = UNSET 1 = SET
6	RO	X	USB_CLKEN: USB AHB clock enable status. Indicates whether the AHB clock to the USB controller is enabled or not. If AHB clock to USB controller is enabled, this bit is set to 1, else it is set to 0. NOTE: even when this is set to 0, all essential blocks that are required to resume USB clocks from suspend will be active and their AHB clock will not be suspended. 0 = UNSET 1 = SET
5	RW	0x0	USB_SUSP_CLR: Suspend Clear Software must write a 1 to this bit to bring the PHY out of suspend mode. This is used when the software stops the PHY clock during suspend and then wants to initiate a resume. Software should also write 0 to clear it. NOTE: It is required that software generate a positive pulse on this bit to guarantee proper operation. 0 = UNSET 1 = SET
4	RW	0x0	USB_WAKE_ON_DISCON_EN_DEV: Wake on Disconnect Enable (device mode) When enabled (1), USB device will wakeup from suspend on a disconnect event. This is only valid when USB controller is in device mode, it is not applicable when USB controller is in host mode. 0 = DISABLE 1 = ENABLE
3	RW	0x0	USB_WAKE_ON_CNNT_EN_DEV: Wake on Connect Enable (device mode) When enabled (1), USB device will wakeup from suspend on a connect event. This is only valid when USB controller is in device mode, it is not applicable when USB controller is in host mode. 0 = DISABLE 1 = ENABLE
2	RW	0x0	USB_WAKE_ON_RESUME_EN: Wake on resume enable If this bit is enabled, USB will wakeup from suspend whenever a resume event is detected on USB. This is valid for both USB device and USB host modes. 0 = DISABLE 1 = ENABLE
1	RW	0x0	USB_WAKEUP_INT_ENB: USB wakeup interrupt enable If this bit is enabled, interrupt is generated whenever USB wakeup event is generated. 0 = DISABLE 1 = ENABLE
0	RO	0x0	USB_WAKEUP_INT_STS: USB wakeup interrupt status This bit is set whenever USB wakes up from suspend (a wakeup event is generated). Software must write a 1 to clear this bit. Note that during the wakeup sequence, PHY clocks will be resumed from suspend. Software can check when the PHY clocks are resumed by reading the bit USB_PHY_CLK_VALID. There is also a separate interrupt generated when PHY clock is resumed if USB_PHY_CLK_VALID_INT_EN is set. During the wakeup sequence, first USB_WAKEUP_INT_STS will be set, and it will take some time for the PHY clock to resume, which can be detected by checking USB_PHY_CLK_VALID. 0 = UNSET 1 = SET

26.5.4.2 USB2_IF_USB_ULPIS2S_CTRL_0

ULPI NULL PHY control register. This register is used to setup parameters for ULPI null PHY mode.

Offset: 418h | Read/Write: R/W | Reset: 0b00000000xxxx0000

Bit	Reset	Description
15	0x0	ULPIS2S_DISABLE_STP_PU: When set to 1 and in ULPIS2S mode, the pullup on the STP pin will NOT be active, even if the remote LINK asks to do so. In this case, an external pullup resistor would be required to ensure valid levels when the remote link is not powered.

Bit	Reset	Description
14	0x0	ULPIS2S_SUPPORT_HS_KEEP_ALIVE: When enabled, the PHY will support HS KeepAlive packets. In that case, this would be the only thing that's supported in Opmode3. All other Opmode3 generate packets are not supported under any circumstances. 0 = DISABLE 1 = ENABLE
13	0x0	ULPIS2S_DISCON_DONT_CHECK_SE0: When enabled, the disconnect detection logic will only check that that the other side is 'driving' tri-state. It won't check whether or not the local side is driving SE0. 0 = DISABLE 1 = ENABLE
12	0x0	ULPIS2S_FORCE_ULPI_CLK_OUT: When enabled and ULPIS2S_ENA is ENABLED, the external ULPI_CLOCK pad will always carry the internal 60MHz clock, even if the interface is in shutdown mode. 0 = DISABLE 1 = ENABLE
11:8	0x0	ULPIS2S_SPARE: Reserved bits When enabled, it will when the other side goes to OPMODE1.
3	0x0	ULPIS2S_PLLU_MASTER_BLASTER60: When enabled, the PLLU 60MHz clock will be forced on. 0 = DISABLE 1 = ENABLE
2	0x0	ULPIS2S_SUPPORT_DISCONNECT: When disabled, the PHY will never detect a Disconnect. 0 = DISABLE 1 = ENABLE
1	0x0	ULPIS2S_SLV1_FORCE_DEVICE: When disabled, the slave port that's connected to the pins can be programmed to be host or a device depending on the value of the DpPulldown and DmPulldown bits in the OTG_CTRL ULPI register. When enables, the values of those bits in the OTG_CTRL register is ignored and the port will always behave like a device. 0 = DISABLE 1 = ENABLE
0	0x0	ULPIS2S_ENA: When enabled, the ULPI link interface coming out of the usb2 controller enters a NULL phy with two slaves. As a result the external pins will have a slave ULPI interface. When disabled, the ULPI link interface coming out of the usb2 controller go straight to the pins. 0 = DISABLE 1 = ENABLE

26.5.4.3 USB2_IF_USB_ULPIS2S_SLV1_ID_0

This register controls the product and vendor ID fields for ULPI null PHY presented to external ULPI master.

Offset: 41ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	ULPIS2S_SLV1_VENDOR_ID: PHY vendor_id as seen by external ULPI master
15:0	0x0	ULPIS2S_SLV1_PRODUCT_ID: PHY product_id as seen by external ULPI master

26.5.4.4 USB2_IF_USB_INTER_PKT_DELAY_CTRL_0

Inter packet delay control. This controls the interpacket delay between two consecutive transmit packets in HS mode. This only applies to host mode as device never transmits two packets in a row.

Offset: 420h | Read/Write: R/W | Reset: 0b010010

Bit	Reset	Description
5:0	0x12	IP_DELAY_TX2TX_HS: HS Tx to Tx inter-packet delay. This is valid only for UHSIC PHY. Software should not change this.

26.5.4.5 USB2_UHSIC_PLL_CFG0_0

UHSIC PHY PLL Configuration Register 0

Offset: 800h | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
0	0x0	UHSIC_PLL_SPARE: Reserved

26.5.4.6 USB2_UHSIC_PLL_CFG1_0

UHSIC PLL and PLLU configuration register 1 PLL CONFIGURATION & PARAMETERS

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

Xtal clock -> enters PLLU to generate 12MHz clock -> enters USB_PHY PLL to generate 480/60 MHz clock

The following parameters control the bring-up of the plls:

- Coming out of reset or suspend
  - -> PllUOnState: start pllu_enable_count and pll_lock_count
  - -> Wait ~1us to actually enable the PLLU (pllu_enable_count == ClkXtal * PLLU_ENABLE_DLY_COUNT * 8)
  - -> Wait ~1ms until PLLU is actually stable (pll_lock_count == ClkXtal * PLLU_STABLE_COUNT * 256) => USB_PHY PLL_ENABLE
- Number for a 19.2MHz Xtal clock
  - PLLU_ENABLE_DLY_COUNT[4:0] = 1 * 19.2 / 8 = 2.4 = 3 = 0x03
  - PLLU_STABLE_COUNT[11:0] = 1000 * 19.2 / 256 = 75 = 0x4b
  - XTAL_FREQ_COUNT[11:0] = 2500 * 19.2 / 256 = 187 = 0xbb

UHSIC REGISTER: UHSIC_PLL_CFG1

This register is used to configure the PHY PLL contained in the UHSIC module as well as the PLLU power up and down.

Offset: 804h | Read/Write: R/W | Reset: 0b00011000000011000000

Bit	Reset	Description
18:14	0x3	UHSIC_PLLU_ENABLE_DLY_COUNT: 1 us / (1/19.2MHz) = 19 / 8 = 2.36 = 3
13	0x0	UHSIC_FORCE_PLLU_POWERUP
12	0x0	UHSIC_FORCE_PLLU_POWERDOWN
11:0	0xc0	UHSIC_XTAL_FREQ_COUNT: 2.5ms / (1/19.2MHz) = 48000 / 256 = 187 = 0xBB

26.5.4.7 USB2_UHSIC_HSRX_CFG0_0

UHSIC High speed receive config 0

UHSIC REGISTER: UHSIC_HSRX_CFG0

Offset: 808h | Read/Write: R/W | Reset: 0b00101001110000111000

Bit	Reset	Description
18	0x0	UHSIC_NO_STRIPPING: Do not strip incoming data
17:13	0xa	UHSIC_IDLE_WAIT: Number of cycles of idle to declare IDLE.
12:8	0xe	UHSIC_ELASTIC_OVERRUN_LIMIT
7	0x0	UHSIC_ELASTIC_OVERRUN_DISABLE: Do not declare overrun errors until overflow of FIFO
6:2	0xe	UHSIC_ELASTIC_UNDERRUN_LIMIT
1	0x0	UHSIC_ELASTIC_UNDERRUN_DISABLE: Do not declare underrun errors
0	0x0	UHSIC_PASS_FEEDBACK: Pass through the feedback, do not block it.

26.5.4.8 USB2_UHSIC_HSRX_CFG1_0

UHSIC High speed receive config 1

UHSIC REGISTER: UHSIC_HSRX_CFG1

Offset: 80ch | Read/Write: R/W | Reset: 0b10001000001010010001 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: Nr of delays cells between UH_RX_STROBE and RxStrobeClk in zero cycle path
13:9	0x14	UHSIC_INPUT_FIFO_DEPTH: Depth of the 2-bit wide input FIFO. Maximum depth is 20. Can be tuned
8	0x1	UHSIC_LINE_STATE_RESUME_FAKE_SE0: When enabled, send an SE0 for 2 LS symbols at the end of ResumeK
7	0x0	UHSIC_LINE_STATE_BYPASS: Bypass LineState reclocking logic
6	0x0	UHSIC_EARLY_LINE_STATE_FILTER: Assumes line state filtering table is inclusive, not exclusive
5:1	0x9	UHSIC_HS_SYNC_START_DLY: How long to wait before start of sync launches RxActive
0	0x1	UHSIC_HS_ALLOW_KEEP_ALIVE: Allow Keep Alive packets

26.5.4.9 USB2_UHSIC_TX_CFG0_0

UHSIC transmit config signals

UHSIC REGISTER: UHSIC_TX_CFG0

Offset: 810h | Read/Write: R/W | Reset: 0b1000000000

Bit	Reset	Description
9	0x1	UHSIC_HS_READY_WAIT_FOR_VALID
8	0x0	UHSIC_PACKET_INVERT_DATA: Invert data during a regular packet

Bit	Reset	Description
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 LineState change to resume
4	0x0	UHSIC_SOF_ON_NO_STUFF: Sof when OpMode 3 -- perhaps, when sending controller made packets
3	0x0	UHSIC_SOF_ON_NO_ENCODE: Sof when OpMode 2 -- not likely, for Chirp
2	0x0	UHSIC_NO_STUFFING: No bit stuffing, static programming
1	0x0	UHSIC_NO_ENCODING: No encoding, static programming
0	0x0	UHSIC_NO_SYNC_NO_EOP: Do not sent SYNC or EOP

26.5.4.10 USB2_UHSIC_MISC_CFG0_0

UHSIC miscellaneous configurations

UHSIC REGISTER: UHSIC_MISC_CFG0

Offset: 814h | Read/Write: R/W | Reset: 0b10001001110100011110

Bit	Reset	Description
18	0x1	UHSIC_EXTEND_BK_ACTIVE: Drive buskeeper one cycle longer when going out of IDLE
17:16	0x0	UHSIC_FORCE_XCVRSEL: Value to be forced on XcvrSelect when FORCE_XCVR_MODE is set.
15	0x0	UHSIC_FORCE_XCVR_MODE: 1: Force the values of XcvrSelect via config bits instead of via the controller
14	0x1	UHSIC_SYMMETRIC_CONNECT_DATA: 0: DATA goes high before STROBE goes low and low before STROBE goes high. 1: DATA goes high before STROBE goes low and goes low *after* STROBE goes high.
13	0x0	UHSIC_ASYNC_CONNECT_DATA: 0: DATA keeps setup and hold requirements during CONNECT. 1: DATA moves together with STROBE
12	0x0	UHSIC_LONG_CONNECT_STROBE: 0: STROBE is 2 periods long during connect. 1: STROBE is 3 periods long during connect
11	0x1	UHSIC_ACTIVE_BK_DRIVE_RX: 1: Use RX state (EOP etc) to determine starting time to drive bus keeper instead of waiting for IDLE detection.
10	0x1	UHSIC_ACTIVE_BK_DRIVE_TX: 1: Use TX state to determine starting time to drive bus keeper instead of waiting for IDLE detection.
9	0x1	UHSIC_DETECT_SHORT_IDLE: 0: use 3 edges (negative and positive) to detect a idle state on the line. 1: use 4 edges.
8	0x0	UHSIC_DETECT_SHORT_CONNECT: 0: use 3 edges (negative and positive) to detect a connect state on the line. 1: use 4 edges.
7	0x1	UHSIC_SUSPEND_EXIT_ON_EDGE: Suspend exit requires edge or simply a value...
6:5	0x0	UHSIC_INJECT_ERROR_TYPE: Force error insertion into RX path. (Used for IOBIST.) 0 = DISABLE 1 = BIT_ERR 2 = RX_ERR 3 = BIT_RX_ERR
4:2	0x3	UHSIC_STABLE_COUNT: Number of cycles of crystal clock of signal not changing to consider stable.

Bit	Reset	Description
1	0x1	UHSIC_STABLE_ALL: Determines if all signal need to be stable to not change a config.
0	0x0	UHSIC_COMB_TERMS: Use combinational terminations or synced through CLKXTAL

26.5.4.11 USB2_UHSIC_MISC_CFG1_0

UHSIC miscellaneous configurations

UHSIC REGISTER: UHSIC_MISC_CFG1

Offset: 818h | Read/Write: R/W | Reset: 0b100001100000000010

Bit	Reset	Description
17	0x1	UHSIC_PHY_XTAL_CLOCKEN: Selects whether to enable the crystal clock in the module.
16:15	0x0	UHSIC_OBS_SEL: Select which one of 4 observation vectors is presented on the observation bus
14	0x0	UHSIC_FORCE_IOBIST_CLK_ON: Always enable IoBist CLK60. This would be required when you want to use RX_ERROR_CNT_EN.
13:2	0x600	UHSIC_PLLU_STABLE_COUNT: WRONG! This should be 1ms -> 0x50
1	0x1	UHSIC_RX_ERROR_CNT_CLR: Clear IOBST RxError counter
0	0x0	UHSIC_RX_ERROR_CNT_EN: Enable IOBIST RxError counter when not in IOBIST mode. Allows one to read out the number of errors via JTAG during normal operation

26.5.4.12 USB2_UHSIC_PADS_CFG0_0

Uhsic Pads settings

UHSIC REGISTER: UHSIC_PADS_CFG

Offset: 81ch | Read/Write: R/W | Reset: 0b00000000100010001000100010001000

Bit	Reset	Description
31:24	0x0	UHSIC_HSIC_OPT: Spare config bits
23:20	0x8	UHSIC_TX_SLEWN: Output slew rate (fall time) adjustment
19:16	0x8	UHSIC_TX_SLEWP: Output slew rate (rise time) adjustment
15:12	0x8	UHSIC_TX_RTUNEN: Fine tuned 50 Ohm termination resistor for NMOS driver
11:8	0x8	UHSIC_TX_RTUNEP: Fine tuned 50 Ohm termination resistor for PMOS driver
7:4	0x8	UHSIC_TX_RTERMN: Output impedance adjustment for NMOS driver
3:0	0x8	UHSIC_TX_RTERMP: Output impedance adjustment for PMOS driver

26.5.4.13 USB2_UHSIC_PADS_CFG1_0

Uhsic Pads settings

Offset: 820h | Read/Write: R/W | Reset: 0b0011001111101

Bit	Reset	Description
12	0x0	UHSIC_RPU_STROBE: Enable pull up on IO_STROBE

Bit	Reset	Description
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

26.5.4.14 USB2_UHSIC_CMD_CFG0_0

UHSIC REGISTER: UHSIC_CMD_CFG0

Determine startup behavior.

When AUTO_NEGIOTIATE == 0

- Firmware is supposed to take care of things.

HOST

if Opmode1: don't drive anything (or only enable bus keepers?)

else

Initial power up:

- Asynchronously drive 00

- After a few Xtal clocks, enable bus keepers

Offset: 824h | Read/Write: R/W | Reset: 0b000101

Bit	Reset	Description
5	0x0	UHSIC_PRETEND_CONNECT_DETECT
4	0x0	UHSIC_FORCE_RESET
3	0x0	UHSIC_FORCE_CONNECT: Upon rising value of this bit, force device to send connect. Only useful when AUTO_CONNECT is disabled.
2	0x1	UHSIC_AUTO_CONNECT: As device, automatically send Connect during activation.
1	0x0	UHSIC_FORCE_ACTIVATE: Upon rising value of this bit, instruct state machine to go into activation mode. Only useful when AUTO_ACTIVATE is disabled.
0	0x1	UHSIC_AUTO_ACTIVATE: Upon power up, automatically move to activation mode and start going through connect procedure.

26.5.4.15 USB2_UHSIC_STAT_CFG0_0

UHSIC REGISTER: UHSIC_STAT_CFG0

Offset: 828h | 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

26.5.4.16 USB2_UHSIC_SPARE_CFG0_0

Utmip spare configurations, spare configuration bits

UHSIC REGISTER: UHSIC_SPARE_CFG0

Offset: 82ch | Read/Write: R/W | Reset: 0b11111111111111110000000000000000

Bit	Reset	Description
31:0	-65536	UHSIC_SPARE: Bit 0 : HS_RX_IPG_ERROR_ENABLE Bit 1 : HS_RX_FLUSH_ALAP Bit 2 : FORCE_TRIM_ZERO Bit 7 :4 : RX_DATA_TRIM[3:0] Bit 11:8 : RX_STROBE_TRIM[3:0] Bit 12 : FORCE_BK_ON Bit 13 : BYPASS_INIT_BLOCK bit 14 : FORCE_SM_IDLE

26.5.4.17 USB2_QH_USB2D_QH_EP_0_OUT_0

USB2D Queue Head for OUT endpoint 0

Offset: 1000h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 0 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 0.

26.5.4.18 USB2_QH_USB2D_QH_EP_0_IN_0

USB2D Queue Head for IN endpoint 0

Offset: 1040h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 0. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 0.

26.5.4.19 USB2_QH_USB2D_QH_EP_1_OUT_0

USB2D Queue Head for OUT endpoint 1

Offset: 1080h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 1 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 1.

26.5.4.20 USB2_QH_USB2D_QH_EP_1_IN_0

USB2D Queue Head for IN endpoint 1

Offset: 10c0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 1. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 1.

26.5.4.21 USB2_QH_USB2D_QH_EP_2_OUT_0

USB2D Queue Head for OUT endpoint 2

Offset: 1100h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 2 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 2.

26.5.4.22 USB2_QH_USB2D_QH_EP_2_IN_0

USB2D Queue Head for IN endpoint 2

Offset: 1140h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 2. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 2.

26.5.4.23 USB2_QH_USB2D_QH_EP_3_OUT_0

USB2D Queue Head for OUT endpoint 3

Offset: 1180h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 3 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 3.

26.5.4.24 USB2_QH_USB2D_QH_EP_3_IN_0

USB2D Queue Head for IN endpoint 3

Offset: 11c0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 3. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 3.

26.5.4.25 USB2_QH_USB2D_QH_EP_4_OUT_0

USB2D Queue Head for OUT endpoint 4

Offset: 1200h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 4 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 4.

26.5.4.26 USB2_QH_USB2D_QH_EP_4_IN_0

USB2D Queue Head for IN endpoint 4

Offset: 1240h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 4. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 4.

26.5.4.27 USB2_QH_USB2D_QH_EP_5_OUT_0

USB2D Queue Head for OUT endpoint 5

Offset: 1280h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 5 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 5.

26.5.4.28 USB2_QH_USB2D_QH_EP_5_IN_0

USB2D Queue Head for IN endpoint 5

Offset: 12c0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 5. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 5.

26.5.4.29 USB2_QH_USB2D_QH_EP_6_OUT_0

USB2D Queue Head for OUT endpoint 6

Offset: 1300h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 6 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 6.

26.5.4.30 USB2_QH_USB2D_QH_EP_6_IN_0

USB2D Queue Head for IN endpoint 6

Offset: 1340h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 6. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 6.

26.5.4.31 USB2_QH_USB2D_QH_EP_7_OUT_0

USB2D Queue Head for OUT endpoint 7

Offset: 1380h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 7 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 7.

26.5.4.32 USB2_QH_USB2D_QH_EP_7_IN_0

USB2D Queue Head for IN endpoint 7

Offset: 13c0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 7. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 7.

26.5.4.33 USB2_QH_USB2D_QH_EP_8_OUT_0

USB2D Queue Head for OUT endpoint 8

Offset: 1400h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 8 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 8.

26.5.4.34 USB2_QH_USB2D_QH_EP_8_IN_0

USB2D Queue Head for IN endpoint 8

Offset: 1440h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 8. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 8.

26.5.4.35 USB2_QH_USB2D_QH_EP_9_OUT_0

USB2D Queue Head for OUT endpoint 9

Offset: 1480h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 9 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 9.

26.5.4.36 USB2_QH_USB2D_QH_EP_9_IN_0

USB2D Queue Head for IN endpoint 9

Offset: 14c0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 9. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 9.

26.5.4.37 USB2_QH_USB2D_QH_EP_10_OUT_0

USB2D Queue Head for OUT endpoint 10

Offset: 1500h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 10 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 10.

26.5.4.38 USB2_QH_USB2D_QH_EP_10_IN_0

USB2D Queue Head for IN endpoint 10

Offset: 1540h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 10. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 10.

26.5.4.39 USB2_QH_USB2D_QH_EP_11_OUT_0

USB2D Queue Head for OUT endpoint 11

Offset: 1580h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 11 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 11.

26.5.4.40 USB2_QH_USB2D_QH_EP_11_IN_0

USB2D Queue Head for IN endpoint 11

Offset: 15c0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 11. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 11.

26.5.4.41 USB2_QH_USB2D_QH_EP_12_OUT_0

USB2D Queue Head for OUT endpoint 12

Offset: 1600h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 12 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 12.

26.5.4.42 USB2_QH_USB2D_QH_EP_12_IN_0

USB2D Queue Head for IN endpoint 12

Offset: 1640h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 12. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 12.

26.5.4.43 USB2_QH_USB2D_QH_EP_13_OUT_0

USB2D Queue Head for OUT endpoint 13

Offset: 1680h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 13 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 13.

26.5.4.44 USB2_QH_USB2D_QH_EP_13_IN_0

USB2D Queue Head for IN endpoint 13

Offset: 16c0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 13. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 13.

26.5.4.45 USB2_QH_USB2D_QH_EP_14_OUT_0

USB2D Queue Head for OUT endpoint 14

Offset: 1700h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 14 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 14.

26.5.4.46 USB2_QH_USB2D_QH_EP_14_IN_0

USB2D Queue Head for IN endpoint 14

Offset: 1740h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 14. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 14.

26.5.4.47 USB2_QH_USB2D_QH_EP_15_OUT_0

USB2D Queue Head for OUT endpoint 15

Offset: 1780h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 15 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 15.

26.5.4.48 USB2_QH_USB2D_QH_EP_15_IN_0

USB2D Queue Head for IN endpoint 15

Offset: 17c0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 15. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 15.

26.5.5 USB3 Controller Registers

26.5.5.1 USB2_CONTROLLER_2_USB2D_ID_0

USB2D Identification Register

Offset: 000h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	REVISION: Revision number of the USB controller. This is set to 0x33.
15:8	X	NID: Ones complement version of ID. This field is set to 0xFA.
7:0	X	ID: Configuration number. This field is set to 0x05

26.5.5.2 USB2_CONTROLLER_2_USB2D_HW_HOST_0

USB2D Hardware Host Register

Offset: 008h | Read/Write: RO | Reset: 0bxxxx

Bit	Reset	Description
3:1	X	NPORT: VUSB_HS_NUM_PORT-1: This host controller has only 1 port. So this field will always be 0.
0	X	HC: VUSB_HS_HOST: Indicates support for host mode. Set to 1.

26.5.5.3 USB2_CONTROLLER_2_USB2D_HW_DEVICE_0

USB2D Hardware Device Register

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

Bit	Reset	Description
5:1	X	DEVEP: VUSB_HS_DV_EP: No. of endpoints supported by this device controller. Set to 16. This includes control endpoint 0.
0	X	DC: Device capable: Set to 1 indicating support for device mode.

26.5.5.4 USB2_CONTROLLER_2_USB2D_HW_TXBUF_0

USB2D Hardware TX Buffer Register

Offset: 010h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	TXCHANADD: VUSB_HS_TX_CHAN_ADD: Total no. of address bits for the transmit buffer of each transmit endpoint. Set to 7. Each transmit buffer is 128 words deep.
15:8	X	TXADD: VUSB_HS_TX_ADD: Total no. of address bits for the transmit buffer. Set to 11. The total depth of the transmit buffer is 2048 words.
7:0	X	TCBURST: VUSB_HS_TX_BURST: Maximum burst size supported by the transmit endpoints for data transfers. Set to 8.

26.5.5.5 USB2_CONTROLLER_2_USB2D_HW_RXBUF_0

USB2D RX Buffer HW Parameters Register

Offset: 014h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:8	X	RXADD: VUSB_HS_RX_ADD: Total no. of address bits for the receive buffer. Set to 7. The total depth of the receive buffer is 128 words
7:0	X	RXBURST: VUSB_HS_RX_BURST: Maximum burst size supported by the receive endpoints for data transfers. Set to 8.

26.5.5.6 USB2_CONTROLLER_2_USB2D_CAPLENGTH_0

USB2D Capability Register Length Register

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

Bit	Reset	Description
7:0	X	CAPLENGTH: Indicates which offset to add to the register base address at the beginning of the Operational Register. Set to 0x40.

26.5.5.7 USB2_CONTROLLER_2_USB2D_HCIVERSON_0

USB2D Host Interface Version Number Register

Offset: 102h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	HCIVERSION: Contains a BCD encoding of the EHCI revision number supported by this host controller. The most significant byte of this register represents a major revision and the least significant byte is the minor revision. This host controller supports EHCI revision 1.00.

26.5.5.8 USB2_CONTROLLER_2_USB2D_HCSPARAMS_0

USB2D Host Control Structural Parameters Register

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

Bit	Reset	Description
27:24	X	N_TT: Number of Transaction Translators: indicates the number of embedded transaction translators associated with the USB2.0 host controller. This field is always set to 1 indicating only 1 embedded TT is implemented in this implementation. This is a non-EHCI field to support embedded TT.
23:20	X	N_PTT: Number of Ports per Transaction Translator: indicates the number of ports assigned to each transaction translator within the USB2.0 host controller. Field always equals N_PORTS. This is a non-EHCI field to support embedded TT.
15:12	X	N_CC: Number of Companion Controller: indicates the number of companion controllers. This field is set to 0.
11:8	X	N_PCC: Number of Ports per Companion Controller: indicates the number of ports supported per internal companion controller. This field is set to 0.
4	X	PPC: Port Power Control: indicates whether the host controller implementation includes port power control. 1 = Ports have port power switches 0= Ports do not have port power switches. This field affects the functionality of the port Power field in each port status and control register. This field is set to 1.
3:0	X	N_PORTS: Number of downstream ports. This field specifies the number of physical downstream ports implemented on this host controller. This field is fixed to 1, since this host controller only supports 1 port.

26.5.5.9 USB2_CONTROLLER_2_USB2D_HCCPARAMS_0

USB2D Host Control Capability Parameters Register

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

Bit	Reset	Description
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".

26.5.5.10 USB2_CONTROLLER_2_USB2D_DCIVERSION_0

USB2D Device Interface Version Number Register

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

Bit	Reset	Description
15:0	X	DCIVERSION: The device controller interface conforms to the two-byte BCD encoding of the interface version number contained in this register.

26.5.5.11 USB2_CONTROLLER_2_USB2D_DCCPARAMS_0

USB2D Device Control Capabilities Register

Offset: 124h | Read/Write: RO | Reset: 0bxxxxxxxxx

Bit	Reset	Description
8	X	HC: Host Capable: 1 = This controller is capable of operating as an EHCI compatible USB 2 0 host controller operating as an EHCI compatible USB 2.0 host controller. This field is set to 1.
7	X	DC: Device Capable: 1 = Controller is capable of operating as USB 2.0 device. This field is set to 1.
4:0	X	DEN: Device Endpoint Number: Number of endpoints built into the device controller. This is set to 16.

26.5.5.12 USB2_CONTROLLER_2_USB2D_USBCMD_0

USB2D USB Command Register

Offset: 140h | Read/Write: R/W | Reset: 0b00001000000x1x11x0000000

Bit	R/W	Reset	Description
23:16	RW	0x8	ITC: Interrupt Threshold Control .Read/Write. Default 08h. The system software uses this field to set the maximum rate at which the host/device controller will issue interrupts. ITC contains the maximum interrupt interval measured in micro-frames. Valid values are shown below. Value Maximum Interrupt Interval 00h Immediate (no threshold) 01h 1 micro-frame 02h 2 micro-frames 04h 4 micro-frames 08h 8 micro-frames 10h 16 micro-frames 20h 32 micro-frames 40h 64 micro-frames 0 = IMMEDIATE 2 = ONE_MF 4 = TWO_MF 8 = EIGHT_MF 16 = SIXTEEN_MF 32 = THIRTY_TWO_MF 64 = SIXTY_FOUR_MF
15	RW	0x0	FS2: Bit 2 of Frame List Size.
14	RW	0x0	ATDTW: Add DTD Tripwire. This bit is used as a semaphore when a dTD is added to an active (primed) endpoint. This bit is set and cleared by software and will be cleared by hardware when a hazard exists such that adding a dTD to a primed endpoint may go unnoticed. 0 = CLEAR 1 = SET
13	RW	0x0	SUTW: Setup Tripwire. This bit is used as a semaphore when the 8 bytes of setup data read extracted by the firmware. If the setup lockout mode is off, then there exists a hazard when new setup data arrives and firmware is copying setup data from the QH for a previous setup packet. This bit is set and cleared by software and will be cleared by hardware when a hazard exists. 0 = CLEAR 1 = SET
11	RW	0x1	ASPE: Asynchronous Schedule Park mode Enable. Software uses this bit to enable or disable Park mode. When this bit is one, Park mode is enabled. When this bit is a zero, Park mode is disabled. This field is set to "1" in this implementation. 0 = DISABLE 1 = ENABLE
9:8	RW	0x3	ASP1_ASP0: Asynchronous Schedule Park Mode Count (OPTIONAL) Read/Write. If the Asynchronous Park Capability bit in the HCCPARAMS register is a one, then this field defaults to 3h and is R/W. Otherwise it defaults to zero and is RO. It contains a count of the number of successive transactions the host controller is allowed to execute from a high-speed queue head on the Asynchronous schedule before continuing traversal of the Asynchronous schedule. Valid values are 1h to 3h. Software must not write a zero to this bit when Park Mode Enable is a one as this will result in undefined behavior. This field is set to 3h in this implementation and is Read/Write capable.
7	RO	X	LR: Light Host/Device Controller Reset (OPTIONAL) . Read Only. Not Implemented. This field will always be "0".
6	RW	0x0	IAA: Interrupt on Async Advance Doorbell. When the host controller has evicted all appropriate cached schedule states, it sets the Interrupt on Async Advance status bit in the USBSTS register. If the Interrupt on Sync Advance Enable bit in the USBINTR register is one, then the host controller will assert an interrupt at the next interrupt threshold. The host controller sets this bit to zero after it has set the Interrupt on Sync Advance status bit in the USBSTS register to one. Software should not write a one to this bit when the asynchronous schedule is inactive. Doing so will yield undefined results. This bit is only used in host mode. Writing a one to this bit when device mode is selected will have undefined results. 0 = CLEAR 1 = SET
5	RW	0x0	ASE: Asynchronous Schedule Enable. This bit controls whether the host controller skips processing the Asynchronous Schedule. 0 = Do not process the Asynchronous Schedule. 1 = Use the ASYNCLISTADDR register to access the Asynchronous Schedule. Only the host controller uses this bit.

Bit	R/W	Reset	Description
			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 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

26.5.5.13 USB2_CONTROLLER_2_USB2D_USBSTS_0

USB2D USB Status Register

Offset: 144h | Read/Write: R/W | Reset: 0b00xx00x1x0x0000x0000

Bit	R/W	Reset	Description
19	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
18	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

Bit	R/W	Reset	Description
			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
16	RO	X	NAKI: NAK Interrupt Bit Read Only. This bit is readonly. It is set by hardware when for a particular endpoint both the TX/RX Endpoint NAK bit and the corresponding TX/RX Endpoint NAK Enable bit are set. This bit is automatically cleared by hardware when the all the enabled TX/RX Endpoint NAK bits are cleared. 0 = DISABLE 1 = ENABLE
15	RW	0x0	AS: Asynchronous Schedule Status. This bit reports the current real status of the Asynchronous Schedule. When set to zero the asynchronous schedule status is disabled and if set to one the status is enabled. The Host Controller is not required to immediately disable or enable the Asynchronous Schedule when software transitions the Asynchronous Schedule Enable bit in the USBCMD register. If AS = ASE: 1= Enable Asynchronous Schedule 0= Disable Asynchronous Schedule Only used by the host controller. 0 = DISABLE 1 = ENABLE
14	RW	0x0	PS: Periodic Schedule Status. This bit reports the current real status of the Periodic Schedule. When set to zero the periodic schedule is disabled, and if set to one the status is enabled. The Host Controller is not required to immediately disable or enable the Periodic Schedule when software transitions the Periodic Schedule Enable bit in the USBCMD register. If PS = PSE then: 1 = Periodic Schedule is enabled or 0 = Periodic Schedule is disabled Only used by the host controller. 0 = DISABLE 1 = ENABLE
13	RO	X	RCL: Reclamation. This is a read-only status bit used to detect an empty asynchronous schedule. Only used by the host controller. 0 = DISABLE 1 = ENABLE
12	RW	0x1	HCH: HCHalted. 1 = Default. This bit is a zero whenever the Run/Stop bit is a one. The Host Controller sets this bit to one after it has stopped executing because of the Run/Stop bit being set to 0, either by software or by the Host Controller hardware (e.g. internal error). Only used by the host controller. 0 = UNHALTED 1 = HALTED
10	RW	0x0	ULPI_INT: ULPI Interrupt. This bit is set whenever an interrupt is received from ULPI PHY. Software writes 1 to clear it. 0 = NOT_ULPI_INT 1 = ULPI_INT
8	RW	0x0	SLI: DCSuspend. When a device controller enters a suspend state from an active state, this bit will be set to a 1. The device controller clears the bit upon exiting from a suspend state. Only used by the device controller. 0 = NOTSUSPEND 1 = SUSPENDED
7	RW	0x0	SRI: SOF Received. When the device controller detects a Start Of (micro) Frame, this bit will be set to a one. When a SOF is extremely late, the device controller will automatically set this bit to indicate that an SOF was expected. Therefore, this bit will be set roughly every 1ms in device FS mode and every 125us in HS mode and will be synchronized to the actual SOF that is received. Since device controller is initialized to FS before connect, this bit Will be set at an interval of 1ms during the prelude to the connect and chirp. In host mode, this bit will be set every 125us and can be used by host controller driver as a time base. Software writes a 1 to this bit to clear it. This is a non-EHCI status bit. 0 = SOF_NOT_RCVD 1 = SOF_RCVD
6	RW	0x0	URI: USB Reset Received. When the device controller detects a USB Reset and enters the default state, this bit is set to a 1. Software can write a 1 to this bit to clear the USB Reset Received status bit. Only used by the device controller. 0 = NO_USB_RESET 1 = USB_RESET

Bit	R/W	Reset	Description
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

26.5.5.14 USB2_CONTROLLER_2_USB2D_USBINTR_0

USB2D USB Interrupt Enable Register

Offset: 148h | Read/Write: R/W | Reset: 0b00x0xxxxx0x000000000

Bit	Reset	Description
19	0x0	UPIE: UPIE Interrupt Enable. 1 = USB controller issues an interrupt if UPA bit in USBSTS register transitions. 0 = DISABLE 1 = ENABLE
18	0x0	UAIE: UAIE Interrupt Enable. 1 = USB controller issues an interrupt if UAI bit in USBSTS register transitions. 0 = DISABLE 1 = ENABLE
16	0x0	NAKE: NAK Interrupt Enable. 1 = USB controller issues an interrupt if NAKI bit in USBSTS register transitions. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
10	0x0	ULPIE: ULPI Interrupt Enable. 1 = USB controller issues an interrupt if ULPI_INT bit in USBSTS register transitions. The interrupt is acknowledged by SW by writing a 1 to the ULPI_INT bit. 0 = DISABLE 1 = ENABLE
8	0x0	SLE: Sleep Enable. 1 = Device controller issues an interrupt if DCSuspend bit in USBSTS register transitions. The interrupt is acknowledged by SW by writing a 1 to the DCSuspend bit. Only used by the device controller. 0 = DISABLE 1 = ENABLE
7	0x0	SRE: SOF Received Enable. 1 = Device controller issues an interrupt if SOF Received bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the SOF Received bit. 0 = DISABLE 1 = ENABLE
6	0x0	URE: USB Reset Enable.1 = Device controller issues an interrupt if USB Reset Received bit in USBSTS register = 1 The interrupt is acknowledged by software clearing the USB Reset Received bit. Only used by the device controller. 0 = DISABLE 1 = ENABLE
5	0x0	AAE: Interrupt on Asynchronous Advance Enable. 1 = the host controller issues an interrupt at the next interrupt threshold if Interrupt on Async Advance bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the Interrupt on Async Advance bit. Only used by the host controller. 0 = DISABLE 1 = ENABLE
4	0x0	SEE: System Error Enable. 1 = Host/device controller issues an interrupt if the System Error bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the System Error bit. 0 = DISABLE 1 = ENABLE
3	0x0	FRE: Frame List Rollover Enable. 1 = Host controller issues an interrupt if Frame List Rollover bit in the USBSTS register = 1. The interrupt is acknowledged by software clearing the Frame List Rollover bit. Only used by the host controller. 0 = DISABLE 1 = ENABLE
2	0x0	PCE: Port Change Detect Enable. 1 = Host/device controller issues an interrupt if Port Change Detect bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the Port Change Detect bit. 0 = DISABLE 1 = ENABLE
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

26.5.5.15 USB2_CONTROLLER_2_USB2D_FRINDEX_0

USB2D USB Frame Index Register

Offset: 14ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxx

Bit	Reset	Description
13:0	X	FRINDEX: Frame Index. The value in this register increments at the end of each time frame (micro-frame). Bits [N: 3] are used for the Frame List current index. Each location of the frame list is accessed 8 times (frames or micro-frames) before moving to the next index. The following illustrates values of N based on the value of the Frame List Size field in the USBCMD register, when used in host mode. USBCMD [Frame List Size] Number Elements N 000b (1024) 12 001b (512) 11 010b (256) 10 011b (128) 9 100b (64) 8 101b (32) 7 110b (16) 6 111b (8) 5 In device mode the value is the current frame number of the last frame transmitted. It is not used as an index. In either mode bits 2:0 indicate the current micro-frame.

26.5.5.16 USB2_CONTROLLER_2_USB2D_PERIODICLISTBASE_0

USB2D Host Controller Frame List Base Address Register

Offset: 154h | Read/Write: R/W | Reset: 0b00000000000000000000

Bit	Reset	Description
31:25	0x0	USBADR: Device mode. The upper seven bits of this register represent the device address. After any controller reset or a USB reset, the device address is set to the default address (0). The default address will match all incoming addresses. Software shall reprogram the address after receiving a SET_ADDRESS request.
24	0x0	USBADRA: Device Address Advance. Default=0. When this bit is 0, any writes to USBADR are instantaneous. When this bit is written to a 1 at the same time or before USBADR is written, the write to the USBADR field is staged and held in a hidden register. After an IN occurs on endpoint 0 and is ACKed, USBADR will be loaded from the holding register. Hardware will automatically clear this bit on the following conditions: 1) IN is ACKed to endpoint 0. (USBADR is updated from staging register). 2) OUT/SETUP occur to endpoint 0. (USBADR is not updated). 3) Device Reset occurs (USBADR is reset to 0). Note: After the status phase of the SET_ADDRESS descriptor, the DCD has 2 ms to program the USBADR field. This mechanism will ensure this specification is met when the DCD can not write of the device address within 2ms from the SET_ADDRESS status phase. If the DCD writes the USBADR with USBADRA=1 after the SET_ADDRESS data phase (before the prime of the status phase), the USBADR will be programmed instantly at the correct time and meet the 2ms USB requirement.
31:12	0x0	BASEADR: Host mode: This 32-bit register contains the beginning address of the Periodic Frame List in the system memory. HCD loads this register prior to starting the schedule execution by the Host Controller. The memory structure referenced by this physical memory pointer is assumed to be 4-Kbyte aligned. The contents of this register are combined with the Frame Index Register (FRINDEX) to enable the Host Controller to step through the Periodic Frame List in sequence. Base Address (Low). These bits correspond to memory address signals [31:12], respectively. Only used by the host controller.

26.5.5.17 USB2_CONTROLLER_2_USB2D_ASYNCLISTADDR_0

USB2D Next Asynchronous List Address Register

Offset: 158h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:11	0x0	EPBASE: Device mode. This register contains the address of the top of the endpoint list in system memory. These bits correspond to memory address signals [31:11], respectively. This field will reference a list of up to 32 Queue Heads (QH). Only used by the device controller.
31:5	0x0	ASYBASE: Host mode. This 32-bit register contains the address of the next asynchronous queue head to be executed by the host. Link Pointer Low (LPL). These bits correspond to memory address signals [31:5], respectively. This field may only reference a Queue Head (OH). Only used by the host controller.

26.5.5.18 USB2_CONTROLLER_2_USB2D_ASYNCTTSTS_0

USB2D Asynchronous Buffer Status for Embedded TT Register

Offset: 15ch | Read/Write: R/W | Reset: 0b0x

Bit	R/W	Reset	Description
1	RW	0x0	TTAC: Embedded TT Async Buffers Clear. (Read/Write to set) This field will clear all pending transactions in the embedded TT Async Buffer(s). The clear will take as much time as necessary to clear buffer without interfering with a transaction in progress. TTAC will return to zero after being set by software only after the actual clear occurs.
0	RO	X	TTAS: Embedded TT Async Buffers Status. (Read Only) This read only bit will be 1 if one or more transactions are being held in the embedded TT Async. Buffers. When this bit is a zero, then all outstanding transactions in the embedded TT have been flushed.

26.5.5.19 USB2_CONTROLLER_2_USB2D_BURSTSIZE_0

USB2D Burst Size register

Offset: 160h | Read/Write: R/W | Reset: 0b0000100000001000

Bit	Reset	Description
15:8	0x8	TXPBURST: Programmable TX Burst Length. (Read/Write) This register represents the maximum length of a burst in 32-bit words while moving data from system memory to the USB bus.
7:0	0x8	RXPBURST: Programmable RX Burst Length. (Read/Write) This register represents the maximum length of a burst in 32-bit words while moving data from the USB bus to system memory.

26.5.5.20 USB2_CONTROLLER_2_USB2D_TXFILLTUNING_0

USB2D Transmit fill tuning register

Offset: 164h | Read/Write: R/W | Reset: 0b000010xxx0000000000000

Bit	Reset	Description
21:16	0x2	TXFIFOTHRES: FIFO Burst Threshold. (Read/Write) This register controls the number of data bursts that are posted to the TX latency FIFO in host mode before the packet begins on to the bus. The minimum value is 2 and this value should be a low as possible to maximize USB performance. A higher value can be used in systems with unpredicable latency and/or insufficient bandwidth where the FIFO may underrun because the data transferred from the latency FIFO to USB occurs before it can be replenished from system memory. This value is ignored if the Stream Disable bit in USBMODE register is set.
12:8	0x0	TXSCHHEALTH: Scheduler Health Counter. (Read/Write To Clear) [Default = 0] This register increments when the host controller fails to fill the TX latency FIFO to the level programmed by TXFIFOTHRES before running out of time to send the packet before the next Start-Of-Frame. This health counter measures the number of times this occurs to provide feedback to selecting a proper TXSCHOH. Writing to this register will clear the counter and this counter will max. at 31.
7:0	0x0	TXSCHOH: Scheduler Overhead. (Read/Write) [Default = 0] This register adds an additional fixed offset to the schedule time estimator described above as Tff. As an approximation, the value chosen for this register should limit the number of back-off events captured in the TXSCHHEALTH to less than 10 per second in a highly utilized bus. Choosing a value that is too high for this register is not desired as it can needlessly reduce USB utilization. The time unit represented in this register is 1.267us when a device is connected in High-Speed Mode. The time unit represented in this register is 6.333us when a device is connected in Low/Full Speed Mode

26.5.5.21 USB2_CONTROLLER_2_USB2D_ICUSB_CTRL_0

USB2D ICUSB control register

This register enables and controls the ICUSB FS/LS transceiver.

Offset: 16ch | Read/Write: R/W | Reset: 0b0000

Bit	Reset	Description
3	0x0	IC_ENB1: ICUSB transceiver enable. This bit enables the ICUSB transceiver . To enable the interface, the bits PTS must be set to 11 in the PORTSCx. Writing a '1' to this bit selects the IC_USB interface. 0 = DISABLE 1 = ENABLE
2:0	0x0	IC_VDD1: ICUSB voltage select. It selects which voltage is being supplied to the ICUSB peripheral. 000 -> No voltage 001 -> 1.0V - reserved 010 -> 1.2V - reserved 011 -> 1.5V - reserved 100 -> 1.8V 101 -> 3.0V 110 -> reserved 111 -> reserved The Voltage negotiation should happen between enabling port power (PP) and asserting the run/stop bit in register.

26.5.5.22 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.

CAUTION 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 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 theoperation 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 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 Control register.

Please refer to the ULPI Specification Revision 1.1 for further information on the use of the optional features.

Offset: 170h | Read/Write: R/W | Reset: 0b000xx00000000000xxxxxxxx00000000

Bit	R/W	Reset	Description
31	RW	0x0	ULPI_WAKEUP: ULPI Wakeup. Writing the 1 to this bit will begin the wakeup operation. The bit will automatically transition to 0 after the wakeup is complete. Once this bit is set, the driver can not set it back to 0. Note: The driver must never execute a wakeup and a read/write operation at the same time. 0 = CLEAR 1 = SET
30	RW	0x0	ULPI_RUN: ULPI read/write Run. Writing the 1 to this bit will begin the read/write operation. The bit will automatically transition to 0 after the read/write is complete. Once this bit is set, the driver can not set it back to 0. Note: The driver must never execute a wakeup and a read/write operation at the same time. 0 = CLEAR 1 = SET
29	RW	0x0	ULPI_RD_WR: ULPI read/write control. (0) Read; (1) Write. This bit selects between running a read or write operation. 0 = READ 1 = WRITE
27	RO	X	ULPI_SYNC_STATE: ULPI sync state. (1) Normal Sync. State. (0) In another state (i.e. carkit, serial, low power) This bit represents the state of the ULPI interface. 0 = NOT_NORMAL 1 = NORMAL
26:24	RW	0x0	ULPI_PORT: ULPI PHY port no. This field should be always written as 0.
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.

26.5.5.23 USB2_CONTROLLER_2_USB2D_ENDPTNAK_0

USB2D Endpoint NAK register

Offset: 178h | 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

26.5.5.24 USB2_CONTROLLER_2_USB2D_ENDPTNAK_ENABLE_0

USB2D Endpoint NAK Enable register

Offset: 17ch | 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

26.5.5.25 SB2_CONTROLLER_2_USB2D_PORTSC1_0

USB2D Port Status/Control 1 Register

Offset: 184h | Read/Write: R/W | Reset: 0b000xxx0000000000xxx1xxx0x0xx010x

Bit	R/W	Reset	Description
31:30	RW	0x0	PTS: Parallel transceiver select. This bit is not defined in the EHCI specification. 0 = UTMI 1 = RESERVED 2 = ULPI 3 = ICUSB_SER
29	RW	0x0	STS: 0 = 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
28	RO	X	PTW: Parallel Transceiver Width. Fixed to 0. This bit is not defined in the EHCI specification. 0 = EIGHT_BIT 1 = RESERVED
27:26	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
25	RW	0x0	SRT: Shorten USB Reset Time. Software should never set this to 1.
24	RW	0x0	PFSC: Port Force Full Speed Connect: Writing this bit to a 1b forces the port to connect at Full Speed only. It disables the chirp sequence that allows the port to identify itself as High Speed. This is useful for testing FS configurations with a HS host, hub or device. This bit is not defined in the EHCI specification. 0 = DONT_FORCE_FULL_SPEED 1 = FORCE_FULL_SPEED
23	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. In device mode, the PHY can be put into Low Power Suspend - Clock disable when the device is not running (USBCMD Run/Stop = 0) or the host has signaled suspend (PORTSC SUSPEND = 1). Low power suspend will be cleared automatically when the host has signaled resume. Before forcing a resume from the device, the device 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. This bit is not defined in the EHCI specification. 0 = DISBLE 1 = ENABLE
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

Bit	R/W	Reset	Description
			controller is working in device mode can result in undefined behavior. 0 = DISBLE 1 = ENABLE
21	RW	0x0	WKDS: Wake on Disconnect Enable: Writing this bit to a one enables the port to be sensitive to device disconnects as wake-up events. This field is zero if Port Power(PP) is zero or in device mode. This bit should only be used when operating in Host mode. Writing this bit to 1 while the controller is working in device mode can result in undefined 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 = DISBLE 1 = ENABLE
20	RW	0x0	WKCN: Wake on Connect Enable: Writing this bit to a one enables the port to be sensitive to device connects as wake-up events. This field is zero if Port Power(PP) is zero or in device mode. This bit should only be used when operating in Host mode. Writing this bit to 1 while the controller is working in device mode can result in undefined 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 = DISBLE 1 = ENABLE
19:16	RW	0x0	PTC: Port Test Control: Any other value than zero indicates that the port is operating in test mode. Value Specific Test 0000b Not enabled 0001b J_ STATE 0010b K_STATE 0011b SEQ_NAK 0100b Packet 0101b FORCE_ENABLE 0110b to 1111b Reserved Refer to Chapter 7 of the USB Specification Revision 2.0 for details on each test mode. 0 = NORMAL_OP 1 = TEST_J 2 = TEST_K 3 = TEST_SE0_NAK 4 = TEST_PKT 5 = TEST_FORCE_ENABLE
15:14	RO	X	PIC: Port Indicator Control: This field is not supported in the current implementation. Please use a GPIO if you wish to use Port Indicators.
13	RO	X	PO: Port Owner. Port owner handoff is not implemented in this design, therefore this bit will 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 controller requires port power control switches. This bit represents the current setting of the switch (0=off, 1=on). When power is not available on a port (i.e. PP equals a 0), the port is non-functional and will not report attaches, detaches, etc. When an over-current condition is detected on a powered port and PPC is a one, the PP bit in each affected port may be transitioned by the host controller driver from a one to a zero (removing power from the port). 0 = NOT_POWERED 1 = POWERED
11:10	RO	X	LS: Line state. These bits reflect the current logical levels of the D+ (bit 10) and D- (bit 11) signal lines. The encoding of the bits are: 00b = SE0 01b = J-state 10b = K-state 11b = Undefined The value of this field is undefined if Port Power(PP) is zero in host mode. In host mode, the use of line-state by the host controller driver is not necessary (unlike EHCI), because the port controller state machine and the port routing manage the connection of LS and FS. In device mode, the use of line-state by the device controller driver is not necessary. 0 = SE0 1 = J_STATE 2 = K_STATE 3 = UNDEFINED
9	RO	X	HSP: When the bit is one, the host/device connected to the port is in high-speed mode and if set to zero, the host/device connected to the port is not in a high-speed mode. Note: HSP is redundant with PSPD(27:26). This bit is not defined in the EHCI specification. 0 = NOT_HIGH_SPEED 1 = HIGH_SPEED
8	RW	0x0	PR: This field is zero if Port Power(PP) is zero. In Host Mode: Read/Write. 1=Port is in Reset.

Bit	R/W	Reset	Description
			0=Port is not in Reset. When software writes a one to this bit the bus-reset sequence as defined in the USB Specification Revision 2.0 is started. This bit will automatically change to zero after the reset sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the reset duration is timed in the driver. In Device Mode: This bit is a read only status bit. Device reset from the USB bus is also indicated in the USBSTS register. 0 = NOT_USB_RESET 1 = USB_RESET
7	RO	X	SUSP: Port suspend. 1=Port in suspend state. 0=Port not in suspend state. In Host Mode: Read/Write. Port Enabled Bit and Suspend bit of this register define the port states as follows: Bits [Port Enabled, Suspend] Port State 0x Disable 10 Enable 11 Suspend When in suspend state, downstream propagation of data is blocked on this port, except for port reset. The blocking occurs at the end of the current transaction if a transaction was in progress when this bit was written to 1. In the suspend state, the port is sensitive to resume detection. Note that the bit status does not change until the port is suspended and that there may be a delay in suspending a port if there is a transaction currently in progress on the USB. The host controller will unconditionally set this bit to zero when software sets the Force Port Resume bit to zero. A write of zero to this bit is ignored by the host controller. If host software sets this bit to a one when the port is not enabled (i.e. Port enabled bit is a zero) the results are undefined. This field is zero if Port Power(PP) is zero in host mode. In Device Mode: Read Only. This bit is a read only status bit. 0 = NOT_SUSPEND 1 = SUSPEND
6	RW	0x0	FPR: Force Port Resume. 1= Resume detected/driven on port. 0=No resume (K state) detected/driven on port. In Host Mode: Software sets this bit to one to drive resume signaling. The Host Controller sets this bit to one if a J-to-K transition is detected while the port is in the Suspend state. When this bit transitions to a one because a J-to-K transition is detected, the Port Change Detect bit in the USBSTS register is also set to one. This bit will automatically change to zero after the resume sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the resume duration is timed in the driver. Note that when the Host controller owns the port, the resume sequence follows the defined sequence documented in the USB Specification Revision 2.0. The resume signaling (Full-speed 'K') is driven on the port as long as this bit remains a one. This bit remains a one until the port has switched to the high-speed idle. Writing a zero has no effect because the port controller will time 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

Bit	R/W	Reset	Description
			port state actually changes. There may be a delay in disabling or enabling a port due to other host controller and bus events. When the port is disabled, (0b) downstream propagation of data is blocked except for reset. This field is zero if Port Power(PP) is zero in host mode. In Device Mode: The device port is always enabled. (This bit will be one) 0 = PORT_DISABLED 1 = PORT_ENABLED
1	RW	0x0	CSC: Connect Status Change: 1 =Change in Current Connect Status. 0=No change (default) In Host Mode: Indicates a change has occurred in the port's Current Connect Status. The host/device controller sets this bit for all changes to the port device connect status, even if system software has not cleared an existing connect status change. For example, the insertion status changes twice before system software has cleared the changed condition, hub hardware will be 'setting' an already-set bit (i.e., the bit will remain set). Software clears this bit by writing a one to it. This field is zero if Port Power(PP) is zero in host mode. This bit is undefined in device controller mode. 0 = NO_CHANGE 1 = CHANGE
0	RO	X	CCS: Current Connect Status: In Host Mode: 1=Device is present on port. 0=No device is present (default) This value reflects the current state of the port, and may not correspond directly to the event that caused the Connect Status Change bit (Bit 1) to be set. This field is zero if Port Power(PP) is zero in host mode. In Device Mode: 1=Attached 0=Not Attached (default) A one indicates that the device successfully attached and is operating in either high speed or full speed as indicated by the High Speed Port bit in this register. A zero indicates that the device did not attach successfully or was forcibly disconnected by the software writing a zero to the Run bit in the USBCMD register. It does not state the device being disconnected or suspended. 0 = NOT_CONNECTED 1 = CONNECTED

26.5.5.26 USB2_CONTROLLER_2_USB2D_OTGSC_0

USB2D Status and Control Register

Offset: 1a4h | Read/Write: R/W | Reset: 0b0000000x0000000xxxxxxxxxx000x00

Bit	R/W	Reset	Description
30	RW	0x0	DPIE: Data Pulse Interrupt Enable. Setting this bit enables the Data pulse interrupt. 0 = DISABLE 1 = ENABLE
29	RW	0x0	ONEMSE: 1 millisecond timer Interrupt enable. Setting this bit enables the 1 millisecond timer interrupt. 0 = DISABLE 1 = ENABLE
28	RW	0x0	BSEIE: B Session End Interrupt Enable. Setting this bit enables the B session end interrupt 0 = DISABLE 1 = ENABLE
27	RW	0x0	BSVIE: B Session Valid Interrupt Enable. Setting this bit enables the B session valid interrupt 0 = DISABLE 1 = ENABLE
26	RW	0x0	ASVIE: A Session Valid Interrupt Enable. Setting this bit enables the A session valid interrupt 0 = DISABLE 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

Bit	R/W	Reset	Description
22	RW	0x0	DPIS: Data Pulse Interrupt Status. This bit is set when data bus pulsing occurs on DP or DM. Data bus pulsing is only detected when USBMODE.CM = Host (11) and PORTSC(0).PortPower = Off (0). Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET
21	RW	0x0	ONEMSS: 1 millisecond timer Interrupt Status: This bit is set once every millisecond. Software writes a 1 to clear it. 0 = INT_CLEAR 1 = INT_SET
20	RW	0x0	BSEIS: B Session End Interrupt Status. This bit is set when VBus has fallen below the B session end threshold. Software writes a 1 to clear this bit . 0 = INT_CLEAR 1 = INT_SET
19	RW	0x0	BSVIS: B Session Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the B session valid threshold (0.8 VDC). Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET
18	RW	0x0	ASVIS: A Session Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the A session valid threshold (0.8 VDC). Software writes a one to clear this bit. 0 = INT_CLEAR 1 = INT_SET
17	RW	0x0	AVVIS: A VBus Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the VBus valid threshold (4.4 VDC) on an A device. Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET
16	RW	0x0	IDIS: USB ID Interrupt Status. This bit is set when a change on the ID input has been detected. Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET
14	RO	X	DPS: Data Bus Pulsing Status. A 1 indicates data bus pulsing is being detected on the port. 0 = STS_CLEAR 1 = STS_SET
13	RO	X	ONEMST: 1 millisecond timer toggle. This bit toggles once per millisecond 0 = STS_CLEAR 1 = STS_SET
12	RO	X	BSE: B session End. Indicates VBus is below the B session end threshold 0 = STS_CLEAR 1 = STS_SET
11	RO	X	BSV: B Session Valid. Indicates VBus is above the B session valid threshold 0 = STS_CLEAR 1 = STS_SET
10	RO	X	ASV: A Session Valid. Indicates VBus is above the A session valid threshold 0 = STS_CLEAR 1 = STS_SET
9	RO	X	AVV: A VBus Valid. Indicates VBus is above the A VBus valid threshold 0 = STS_CLEAR 1 = STS_SET
8	RO	X	ID: USB ID: 0 = A-device 1 = B-device 0 = A_DEV 1 = B_DEV
5	RW	0x0	IDPU: USB ID Pullup 0 = CLEAR 1 = SET

Bit	R/W	Reset	Description
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: Termination. This bit must be set when the 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

26.5.5.27 USB2_CONTROLLER_2_USB2D_USBMODE_0

USB2D USB Device Mode Register

Offset: 1a8h | Read/Write: RO | Reset: 0bxxxxx

Bit	Reset	Description
4	X	SDIS: Stream disbable: 1 Streaming is disabled - helpful to avoid overrun/underruns when system load is too high. 0 = STREAM_ENABLE 1 = STREAM_DISABLE
3	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_OFF 1 = LOCKOUT_ON
2	X	ES: Endian Select: Note: For this implementation, this should be always set to 0 (little endian). 0 = LITTLE_ENDIAN 1 = RESERVED
1:0	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

26.5.5.28 USB2_CONTROLLER_2_USB2D_ENDPTSETUPSTAT_0

USB2D Endpoint Setup Status Register

Offset: 1ach | Read/Write: R/W | Reset: 0b0000000000000000

Bit	Reset	Description
15	0x0	ENDPTSETUPSTAT15: Endpoint 15 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
14	0x0	ENDPTSETUPSTAT14: Endpoint 14 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
13	0x0	ENDPTSETUPSTAT13: Endpoint 13 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
12	0x0	ENDPTSETUPSTAT12: Endpoint 12 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
11	0x0	ENDPTSETUPSTAT11: Endpoint 11 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
10	0x0	ENDPTSETUPSTAT10: Endpoint 10 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
9	0x0	ENDPTSETUPSTAT9: Endpoint 9 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
8	0x0	ENDPTSETUPSTAT8: Endpoint 8 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD

Bit	Reset	Description
		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
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

26.5.5.29 USB2_CONTROLLER_2_USB2D_ENDPTPRIME_0

USB2D Endpoint Initialization Register

Offset: 1b0h | 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 endpiont. 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 endpiont. 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 endpiont. 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 endpiont. 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 endpiont. 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 endpiont. 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 endpiont. 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 endpiont. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME

Bit	Reset	Description
		1 = PRIME
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 endpiont. 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 endpiont. 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 endpiont. 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 endpiont. 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 endpiont. 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 endpiont. 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 endpiont. 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 endpiont. 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

Bit	Reset	Description
		to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
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
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

Bit	Reset	Description
		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
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

26.5.5.30 USB2_CONTROLLER_2_USB2D_ENDPTFLUSH_0

USB2D Endpoint De-Initialization Register

Offset: 1b4h | 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
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

Bit	Reset	Description
		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
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

Bit	Reset	Description
		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
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

Bit	Reset	Description
		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

26.5.5.31 USB2_CONTROLLER_2_USB2D_ENDPTSTATUS_0

USB2D Endpoint Status Register

Offset: 1b8h | 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY

Bit	Reset	Description
		1 = READY
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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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

Bit	Reset	Description
		ENDPTPRIME register and endpoint indicating ready. This delay ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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
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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by

Bit	Reset	Description
		the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by 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 ime varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY

26.5.5.32 USB2_CONTROLLER_2_USB2D_ENDPTCOMPLETE_0

USB2D Endpoint Complete Register

Offset: 1bch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ETCE15: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
30	0x0	ETCE14: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
29	0x0	ETCE13: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
28	0x0	ETCE12: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
27	0x0	ETCE11: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
26	0x0	ETCE10: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) 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

Bit	Reset	Description
		1 = COMPLETE
22	0x0	ETCE6: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
21	0x0	ETCE5: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
20	0x0	ETCE4: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
19	0x0	ETCE3: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
18	0x0	ETCE2: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
17	0x0	ETCE1: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
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

Bit	Reset	Description
		USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
12	0x0	ERCE12: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
11	0x0	ERCE11: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
10	0x0	ERCE10: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
9	0x0	ERCE9: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
8	0x0	ERCE8: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
7	0x0	ERCE7: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 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

Bit	Reset	Description
		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

26.5.5.33 USB2_CONTROLLER_2_USB2D_ENDPTCTRL0_0

USB2D Endpoint Control 0 Register

Offset: 1c0h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23	X	TXE: TX Endpoint Enable. Endpoint 0 is always enabled. 0 = DISABLE 1 = ENABLE
19:18	X	TXT: TX Endpoint Type. Endpoint0 is fixed as a Control Endpoint. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
16	X	TXS: TX Endpoint Stall: Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request. 0 = EP_OK 1 = EP_STALL
7	X	RXE: RX Endpoint Enable. Endpoint 0 is always enabled. 0 = DISABLE 1 = ENABLE
3:2	X	RXT: RX Endpoint Type. Endpoint 0 is fixed as a Control Endpoint. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
0	X	RXS: RX Endpoint Stall: Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request. 0 = EP_OK 1 = EP_STALL

26.5.5.34 USB2_CONTROLLER_2_USB2D_ENDPTCTRL1_0

USB2D Endpoint Control 1 Register

Offset: 1c4h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above, 0 = EP_OK

Bit	R/W	Reset	Description
			1 = EP_STALL

26.5.5.35 USB2_CONTROLLER_2_USB2D_ENDPTCTRL2_0

USB2D Endpoint Control 2 Register

Offset: 1c8h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.

Bit	R/W	Reset	Description
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

26.5.5.36 USB2_CONTROLLER_2_USB2D_ENDPTCTRL3_0

USB2D Endpoint Control 3 Register

Offset: 1cch | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL

Bit	R/W	Reset	Description
			1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

26.5.5.37 USB2_CONTROLLER_2_USB2D_ENDPTCTRL4_0

USB2D Endpoint Control 4 Register

Offset: 1d0h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.

Bit	R/W	Reset	Description
			0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

26.5.5.38 USB2_CONTROLLER_2_USB2D_ENDPTCTRL5_0

USB2D Endpoint Control 5 Register

Offset: 1d4h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING

Bit	R/W	Reset	Description
			1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

26.5.5.39 USB2_CONTROLLER_2_USB2D_ENDPTCTRL6_0

USB2D Endpoint Control 6 Register

Offset: 1d8h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE

Bit	R/W	Reset	Description
			1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

26.5.5.40 USB2_CONTROLLER_2_USB2D_ENDPTCTRL7_0

USB2D Endpoint Control 7 Register

Offset: 1dch | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above.

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 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

26.5.5.41 USB2_CONTROLLER_2_USB2D_ENDPTCTRL8_0

USB2D Endpoint Control 8 Register

Offset: 1e0h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.

Bit	R/W	Reset	Description
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This 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

26.5.5.42 USB2_CONTROLLER_2_USB2D_ENDPTCTRL9_0

USB2D Endpoint Control 9 Register

Offset: 1e4h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL

Bit	R/W	Reset	Description
			1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

26.5.5.43 USB2_CONTROLLER_2_USB2D_ENDPTCTRL10_0

USB2D Endpoint Control 10 Register

Offset: 1e8h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.

Bit	R/W	Reset	Description
			0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

26.5.5.44 USB2_CONTROLLER_2_USB2D_ENDPTCTRL11_0

USB2D Endpoint Control 11 Register

Offset: 1ech | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING

Bit	R/W	Reset	Description
			1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

26.5.5.45 USB2_CONTROLLER_2_USB2D_ENDPTCTRL12_0

USB2D Endpoint Control 12 Register

Offset: 1f0h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK

Bit	R/W	Reset	Description
			1 = EP_STALL

26.5.5.46 USB2_CONTROLLER_2_USB2D_ENDPTCTRL13_0

USB2D Endpoint Control 13 Register

Offset: 1f4h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.

Bit	R/W	Reset	Description
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

26.5.5.47 USB2_CONTROLLER_2_USB2D_ENDPTCTRL14_0

USB2D Endpoint Control 14 Register

Offset: 1f8h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL

Bit	R/W	Reset	Description
			1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

26.5.5.48 USB2_CONTROLLER_2_USB2D_ENDPTCTRL15_0

USB2D Endpoint Control 15 Register

Offset: 1fch | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.

Bit	R/W	Reset	Description
			0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

26.5.6 USB3 Controller Interface Registers

Interface registers for USB3 controller. These are used to generate the actual RTL registers for USB3 controller interface.

26.5.6.1 USB3_IF_USB_SUSP_CTRL_0

USB suspend control register. This register controls the suspend and reusume behavior of USB controller/PHY.

Offset: 400h | Read/Write: R/W | Reset: 0b0000x000000xx000000

Bit	R/W	Reset	Description
18:16	RW	0x0	USB_WAKEUP_DEBOUNCE_COUNT: USB PHY wakeup debounce counter USB will debounce any wakeup event by the number of clocks programmed in this counter. A value of 0 results in no debounce.
15	RW	0x0	ICUSB_MOD_CLK_ENB: ICUSB module clock enable. Enables transceiver clock to USB controller when in ICUSB mode. After setting ICUSB_PHY_ENB, software needs to wait until PLLU output is stable before setting ICUSB_MOD_CLK_ENB to ENABLE. This will be reset to DISABLE whenever ICUSB is in suspend. Software needs to enable it after ICUSB comes out of suspend after waiting for PLLU output to be stable again. 0 = DISABLE 1 = ENABLE
13	RW	0x0	ICUSB_PHY_ENB: Enable ICUSB PHY mode Setting this will enable the PLLU output clock when ICUSB is not in suspend 0 = DISABLE 1 = ENABLE
12	RW	0x0	UTMIP_PHY_ENB: Enable UTMIP PHY mode Set this to 1 if using UTMIP PHY. Otherwise set this to 0. 0 = DISABLE 1 = ENABLE
11	RW	0x0	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

Bit	R/W	Reset	Description
9	RW	0x0	USB_PHY_CLK_VALID_INT_ENB: USB PHY clock valid interrupt enable If this bit is enabled, interrupt is generated whenever USB clocks are resumed from a suspend. 0 = DISABLE 1 = ENABLE
8	RO	0x0	USB_PHY_CLK_VALID_INT_STS: USB PHY clock valid interrupt status This bit is set whenever USB PHY clock is waked up from suspend. Software must write a 1 to clear this bit. 0 = UNSET 1 = SET
7	RO	X	USB_PHY_CLK_VALID: USB PHY clock valid status This bit indicates whether the USB PHY is generating a valid clock to the USB controller. If USB PHY clock is running, this bit is set to 1, else it is set to 0. 0 = UNSET 1 = SET
6	RO	X	USB_CLKEN: USB AHB clock enable status. Indicates whether the AHB clock to the USB controller is enabled or not. If AHB clock to USB controller is enabled, this bit is set to 1, else it is set to 0. NOTE: even when this is set to 0, all essential blocks that are required to resume USB clocks from suspend will be active and their AHB clock will not be suspended. 0 = UNSET 1 = SET
5	RW	0x0	USB_SUSP_CLR: Suspend Clear Software must write a 1 to this bit to bring the PHY out of suspend mode. This is used when the software stops the PHY clock during suspend and then wants to initiate a resume. Software should also write 0 to clear it. NOTE: It is required that software generate a positive pulse on this bit to guarantee proper operation. 0 = UNSET 1 = SET
4	RW	0x0	USB_WAKE_ON_DISCON_EN_DEV: Wake on Disconnect Enable (device mode) When enabled (1), USB device will wakeup from suspend on a disconnect event. This is only valid when USB controller is in device mode, it is not applicable when USB controller is in host mode. 0 = DISABLE 1 = ENABLE
3	RW	0x0	USB_WAKE_ON_CNNT_EN_DEV: Wake on Connect Enable (device mode) When enabled (1), USB device will wakeup from suspend on a connect event. This is only valid when USB controller is in device mode, it is not applicable when USB controller is in host mode. 0 = DISABLE 1 = ENABLE
2	RW	0x0	USB_WAKE_ON_RESUME_EN: Wake on resume enable If this bit is enabled, USB will wakeup from suspend whenever a resume event is detected on USB. This is valid for both USB device and USB host modes. 0 = DISABLE 1 = ENABLE
1	RW	0x0	USB_WAKEUP_INT_ENB: USB wakeup interrupt enable If this bit is enabled, interrupt is generated whenever USB wakeup event is generated. 0 = DISABLE 1 = ENABLE
0	RO	0x0	USB_WAKEUP_INT_STS: USB wakeup interrupt status This bit is set whenever USB wakes up from suspend (a wakeup event is generated). Software must write a 1 to clear this bit. Note that during the wakeup sequence, PHY clocks will be resumed from suspend. Software can check when the PHY clocks are resumed by reading the bit USB_PHY_CLK_VALID. There is also a separate interrupt generated when PHY clock is resumed if USB_PHY_CLK_VALID_INT_EN is set. During the wakeup sequence, first USB_WAKEUP_INT_STS will be set, and it will take some time for the PHY clock to resume, which can be detected by checking USB_PHY_CLK_VALID. 0 = UNSET 1 = SET

26.5.6.2 USB3_IF_USB_PHY_VBUS_SENSORS_0

USB PHY VBUS SENSORS control register. This register controls the VBUS sensors in the USB PHY.

Tegra 2 Series devices include the following 4 VBUS sensors:

1. A_VBUS_VLD
2. A_SESS_VLD
3. B_SESS_VLD
4. B_SESS_END

The debounced status of each sensor can be read from _STS. The field _CHG_DET is set to 1 whenever a change is detected in the value. If _INT_EN is set, then an interrupt is generated to the processor. This interrupt can be routed to CPU/COP by appropriately writing the USBD bits in the interrupt controller registers.

In case Software wants to override the value for a , it can set _SW_EN to 1, and set _SW_VALUE to 1 or 0 as per the requirement.

We have two debouncers for each sensor - DEBOUNCE_A and DEBOUNCE_B. The debounce values for them are controlled by the register UTMIP_DEBOUNCE_CFG0, fields UTMIP_BIAS_DEBOUNCE_A and UTMIP_BIAS_DEBOUNCE_B. For each sensor, we can select whether to use DEBOUNCE_A or DEBOUNCE_B by setting the field _DEB_SEL_B to apprprpriate 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.

The source for A_SESS_VLD sensor can be programmed according to the setting of USB1_VBUS_SENSE_CTL in register USB1_LEGACY_CTRL.

Offset: 404h | Read/Write: R/W | Reset: 0b0000x00x0000x00x0000x00x0000x00

Bit	R/W	Reset	Description
30	RW	0x0	A_VBUS_VLD_WAKEUP_EN: A_VBUS_VLD wakeup enable If this bit is enabled, USB will wakeup from suspend whenever a change is detected on A_VBUS_VLD. 0 = DISABLE 1 = ENABLE
29	RW	0x0	A_VBUS_VLD_DEB_SEL_B: A_VBUS_VLD debounce A/B select Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
28	RW	0x0	A_VBUS_VLD_SW_VALUE: A_VBUS_VLD software value Software should write the appropriate value (1/0) to set/unset the A_VBUS_VLD status. This is only valid when A_VBUS_VLD_SW_EN is set. 0 = UNSET 1 = SET
27	RW	0x0	A_VBUS_VLD_SW_EN: A_VBUS_VLD software enable Enable Software Controlled A_VBUS_VLD. Software sets this bit to drive the value in A_VBUS_VLD_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
26	RO	X	A_VBUS_VLD_STS: A_VBUS_VLD status This is set to 1 whenever A_VBUS_VLD sensor output is 1. 0 = UNSET 1 = SET
25	RO	0x0	A_VBUS_VLD_CHG_DET: A_VBUS_VLD change detect. This field is set by hardware whenever a change is detected in the value of A_VBUS_VLD. software writes a 1 to clear it 0 = UNSET 1 = SET
24	RW	0x0	A_VBUS_VLD_INT_EN: A_VBUS_VLD interrupt enable If this field is set to 1, an interrupt is generated whenever A_VBUS_VLD_CHG_DET is set to 1. 0 = DISABLE

Bit	R/W	Reset	Description
			1 = ENABLE
22	RW	0x0	A_SESS_VLD_WAKEUP_EN: A_SESS_VLD wakeup enable If this bit is enabled, USB will wakeup from suspend whenever a change is detected on A_SESS_VLD. 0 = DISABLE 1 = ENABLE
21	RW	0x0	A_SESS_VLD_DEB_SEL_B: A_SESS_VLD debounce A/B select Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
20	RW	0x0	A_SESS_VLD_SW_VALUE: A_SESS_VLD software value Software should write the appropriate value (1/0) to set/unset the A_SESS_VLD status. This is only valid when A_SESS_VLD_SW_EN is set. 0 = UNSET 1 = SET
19	RW	0x0	A_SESS_VLD_SW_EN: A_SESS_VLD software enable Enable Software Controlled A_SESS_VLD. Software sets this bit to drive the value in A_SESS_VLD_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
18	RO	X	A_SESS_VLD_STS: A_SESS_VLD status This is set to 1 whenever A_SESS_VLD sensor output is 1. 0 = UNSET 1 = SET
17	RO	0x0	A_SESS_VLD_CHG_DET: A_SESS_VLD change detect. This field is set by hardware whenever a change is detected in the value of A_SESS_VLD. software writes a 1 to clear it 0 = UNSET 1 = SET
16	RW	0x0	A_SESS_VLD_INT_EN: A_SESS_VLD interrupt enable If this field is set to 1, an interrupt is generated whenever A_SESS_VLD_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
14	RW	0x0	B_SESS_VLD_WAKEUP_EN: B_SESS_VLD wakeup enable If this bit is enabled, USB will wakeup from suspend whenever a change is detected on B_SESS_VLD. 0 = DISABLE 1 = ENABLE
13	RW	0x0	B_SESS_VLD_DEB_SEL_B: B_SESS_VLD debounce A/B select Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
12	RW	0x0	B_SESS_VLD_SW_VALUE: B_SESS_VLD software value Software should write the appropriate value (1/0) to set/unset the B_SESS_VLD status. This is only valid when B_SESS_VLD_SW_EN is set. 0 = UNSET 1 = SET
11	RW	0x0	B_SESS_VLD_SW_EN: B_SESS_VLD software enable Enable Software Controlled B_SESS_VLD. Software sets this bit to drive the value in B_SESS_VLD_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
10	RO	X	B_SESS_VLD_STS: B_SESS_VLD status This is set to 1 whenever B_SESS_VLD sensor output is 1. 0 = UNSET 1 = SET
9	RO	0x0	B_SESS_VLD_CHG_DET: B_SESS_VLD change detect. This field is set by hardware whenever a change is detected in the value of B_SESS_VLD. software writes a 1 to clear it

Bit	R/W	Reset	Description
			0 = UNSET 1 = SET
8	RW	0x0	B_SESS_VLD_INT_EN: B_SESS_VLD interrupt enable If this field is set to 1, an interrupt is generated whenever B_SESS_VLD_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
6	RW	0x0	B_SESS_END_WAKEUP_EN: B_SESS_END wakeup enable If this bit is enabled, USB will wakeup from suspend whenever a change is detected on B_SESS_END. 0 = DISABLE 1 = ENABLE
5	RW	0x0	B_SESS_END_DEB_SEL_B: B_SESS_END debounce A/B select Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
4	RW	0x0	B_SESS_END_SW_VALUE: B_SESS_END software value Software should write the appropriate value (1/0) to set/unset the B_SESS_END status. This is only valid when B_SESS_END_SW_EN is set. 0 = UNSET 1 = SET
3	RW	0x0	B_SESS_END_SW_EN: B_SESS_END software enable Enable Software Controlled B_SESS_END Software sets this bit to drive the value in B_SESS_END_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
2	RO	X	B_SESS_END_STS: B_SESS_END status This is set to 1 whenever B_SESS_END sensor output is 1. 0 = UNSET 1 = SET
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

26.5.6.3 USB3_IF_USB_PHY_VBUS_WAKEUP_ID_0

USB PHY VBUS wakeup and ID control register. This register controls the battery charger (VDAT_DET), VBUS_WAKEUP and ID sensors.

The following sensors are in this register:

1. VBUS_WAKEUP

2. ID

3. VDAT_DET

the debounced status of each sensor can be read from _STS. The field _CHG_DET is set to 1 whenever a change is detected in the value of

. If _INT_EN is set, then an interrupt is generated to the processor. This interrupt can be routed to CPU/COP by appropriately writing the USBD bits in the interrupt controller registers.

In case Software wants to override the value for a , it can set _SW_EN to 1, and set _SW_VALUE to 1 or 0 as per the requirement.

We have two debouncers for each sensor - DEBOUNCE_A and DEBOUNCE_B. The debounce values for them are controlled by the register UTMIP_DEBOUNCE_CFG0, fields UTMIP_BIAS_DEBOUNCE_A and UTMIP_BIAS_DEBOUNCE_B. For each sensor, we can select whether to use DEBOUNCE_A or DEBOUNCE_B by setting the field _DEB_SEL_B to apprORpriate value (SEL_A or SEL_B).

Note: Do not set either UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B to 0x0. If not using one of the debouncers, keep it at the default value of 0xFFFF.

There is no debouncer for VDAT_DET.

Offset: 408h | Read/Write: R/W | Reset: 0b0xxxxxxx000x00xx000x00x1000x00

Bit	R/W	Reset	Description
30	RW	0x0	VBUS_WAKEUP_WAKEUP_EN: VBUS_WAKEUP wakeup enable If this bit is enabled, USB will wakeup from suspend whenever a change is detected on VBUS_WAKEUP. 0 = DISABLE 1 = ENABLE
21	RW	0x0	VDAT_DET_DEB_SEL_B: VDAT_DET 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	VDAT_DET_SW_VALUE: VDAT_DET software value Software should write the appropriate value (1/0) to set/unset the VDAT_DET status. This is only valid when VDAT_DET_SW_EN is set. 0 = UNSET 1 = SET
19	RW	0x0	VDAT_DET_SW_EN: VDAT_DET software enable Enable Software Controlled VDAT_DET. Software sets this bit to drive the value in VDAT_DET_SW_VALUE to the USB controller 0 = DISABLE 1 = ENABLE
18	RO	X	VDAT_DET_STS: VDAT_DET status This is set to 1 whenever VDAT_DET sensor output is 1. 0 = UNSET 1 = SET
17	RO	0x0	VDAT_DET_CHG_DET: VDAT_DET change detect. This field is set by hardware whenever a change is detected in the value of VDAT_DET. software writes a 1 to clear it 0 = UNSET 1 = SET
16	RW	0x0	VDAT_DET_INT_EN: VDAT_DET interrupt enable If this field is set to 1, an interrupt is generated whenever VDAT_DET_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
13	RW	0x0	VBUS_WAKEUP_DEB_SEL_B: VBUS_WAKEUP debounce A/B select Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
12	RW	0x0	VBUS_WAKEUP_SW_VALUE: VBUS wakeup software value Software should write the appropriate value (1/0) to set/unset the VBUS_WAKEUP status. This is only valid when VBUS_WAKEUP_SW_EN is set. 0 = UNSET 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.

Bit	R/W	Reset	Description
			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

26.5.6.4 USB3_IF_USB_PHY_ALT_VBUS_STS_0

USB PHY Alternate VBUS status register

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

Bit	Reset	Description
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

26.5.6.5 USB3_IF_ICUSB_XCVR_CFG_0

ICUSB Transceiver Configuration register

Offset: 414h | Read/Write: R/W | Reset: 0bxxxxxxxxxxx10000xx0000000xxx0111

Bit	R/W	Reset	Description
31:24	RO	X	ICUSB_CALOUT: ICUSB PHY calibration code
20	RW	0x1	ICUSB_CALOUT_EN: ICUSB PHY auto-calibration enable 0 = DISABLE 1 = ENABLE
19:16	RW	0x0	ICUSB_DRV: ICUSB PHY Drive strength offset
13:8	RW	0x0	ICUSB_SLEW: ICUSB PHY FS/LS slew rate control
7	RW	0x0	ICUSB_SEL_DIFF_RCVR: ICUSB differential receiver select 0 = SINGLE 1 = DIFF
3	RW	0x0	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

Bit	R/W	Reset	Description
			1 = OFF
0	RW	0x1	ICUSB_PD_TX: ICUSB PHY Low/full-speed driver power down 0 = NORMAL 1 = OFF

26.5.6.6 USB3_IF_USB_INTER_PKT_DELAY_CTRL_0

Inter packet delay control. This controls the interpacket delay between two consecutive transmit packets in HS mode. This only applies to host mode as device never transmits two packets in a row.

Offset: 420h | Read/Write: R/W | Reset: 0b010010

Bit	Reset	Description
5:0	0x12	IP_DELAY_TX2TX_HS: HS Tx to Tx inter-packet delay. This is valid only for UTMIP PHY Software should not change this.

26.5.6.7 USB3_UTMIP_PLL_CFG0_0

USB_PHY PLL Configuration Register 0 The data sampling frequency relies on a 960MHz clock, so the goal of the PLL is to have: In_Frequency*(PLL_VCOMULTBY2+1)*(PLL_NDIV/PLL_MDIV) = 960MHz With a 12MHz input from PLL_U, the default setting of PLL_VCOMULTBY2=1, PLL_NDIV=40 and PLL_MDIV=1 results in a correct output.

Offset: 800h | Read/Write: R/W | Reset: 0b00000000001010000000000110000000

Bit	Reset	Description
30:28	0x0	UTMIP_PLL_SELECT: Selects which of the eight 60MHz clock phases to produce at the output of the USB PHY PLL. This is for a test mode. See cell specification.
27	0x0	UTMIP_PLL_PDIVRST: PDIVRST of the UTMIP PHY PLL. Reserved. Keep at 0x0. Currently there is no post divider on this PLL. See cell specification.
26:24	0x0	UTMIP_PLL_PDIV: PDIV[2:0] input of the USB_PHY PLL. Reserved. Keep at 0x0. Currently there is no post divider on this PLL. See cell specification.
23:16	0x28	UTMIP_PLL_NDIV: NDIV[7:0] input of USB_PHY PLL. This is the feedback divider on the VCO feedback. 0x0 is not allowed. See cell specification.
15:8	0x1	UTMIP_PLL_MDIV: MDIV[7:0] input of the USB_PHY PLL. This is the predivide on the PLL. 0x0 is not allowed. See cell specification.
7	0x1	UTMIP_PLL_VCOMULTBY2: VCOMULTBY2 control of the UTMIP PHY PLL. Recommended setting is on. Additional divide by 2 on the VCO feedback. Which is setting the bit to 1. See cell spec.
6:1	0x0	UTMIP_PLL_LOCKSEL: LOCKSEL[5:0] input of USB_PHY PLL. Used in test modes only. See cell specification.
0	0x0	UTMIP_PLL_LOCKENABLE: LOCK_ENABLE input of USB_PHY PLL. Normally only used during test. See cell specification.

26.5.6.8 USB3_UTMIP_PLL_CFG1_0

UTMIP PLL and PLLU configuration register 1

PLL CONFIGURATION & PARAMETERS

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

Xtal clock -> enters PLL_U to generate 12MHz clock -> enters USB_PHY PLL to generate 480/60 MHz clock

The following parameters control the bringup of the plls:

Coming out of reset or suspend

- -> PllUOnState: start pllu_enable_count and pll_lock_count
- -> Wait ~1us to actually enable the PLL_U (pllu_enable_count == ClkXtal * PLLU_ENABLE_DLY_COUNT * 8)
- -> Wait ~1ms until PLL_U is actually stable (pll_lock_count == ClkXtal * PLLU_STABLE_COUNT * 256) => USB_PHY PLL_ENABLE
- pll_active_count = 0
- -> Wait 5us to enable pll_active: pll_active_count == ClkXtal * PLL_ACTIVE_DLY_COUNT * 16 => USB_PHY PLL_ACTIVE
- -> Wait ~2.5ms until USB_PHY is actually stable (pll_lock_count == ClkXtal * XTAL_FREQ_COUNT * 256) => USB_PHY

Numbers for a 19.2MHz Xtal clock

- PLLU_ENABLE_DLY_COUNT[4:0] = 1 * 19.2 / 8 = 2.4 = 3 = 0x03
- PLLU_STABLE_COUNT[11:0] = 1000 * 19.2 / 256 = 75 = 0x4b PLL_ACTIVE_DLY_COUNT[4:0] = 5 * 19.2 / 16 = 6 = 0x06
- XTAL_FREQ_COUNT[11:0] = 2500 * 19.2 / 256 = 187 = 0xbb

Offset: 804h | Read/Write: R/W | Reset: 0b00011000001000000000000011000000

Bit	Reset	Description
31:27	0x3	UTMIP_PLLU_ENABLE_DLY_COUNT: Controls the wait time to enable PLL_U when coming out of suspend or reset.
26:18	0x8	UTMIP_PLL_SETUP: SETUP[8:0] input of USB_PHY PLL.
17	0x0	UTMIP_FORCE_PLLU_POWERUP: Force PLL_U into power up.
16	0x0	UTMIP_FORCE_PLLU_POWERDOWN: Force PLL_U into power down. (Overrides FORCE_PLLU_POWERUP.)
15	0x0	UTMIP_FORCE_PLL_ENABLE_POWERUP: Force USB_PHY PLL pll_enable input on.
14	0x0	UTMIP_FORCE_PLL_ENABLE_POWERDOWN: Force USB_PHY PLL pll_enable input off. (Overrides FORCE_PLL_ENABLE_POWERUP.)
13	0x0	UTMIP_FORCE_PLL_ACTIVE_POWERUP: Force USB_PHY PLL pll_active input on.
12	0x0	UTMIP_FORCE_PLL_ACTIVE_POWERDOWN: Force USB_PHY PLL pll_active input off. (Overrides FORCE_PLL_ACTIVE_POWERUP.)
11:0	0xc0	UTMIP_XTAL_FREQ_COUNT: Determines the time to wait until the output of USB_PHY PLL is considered stable.

26.5.6.9 USB3_UTMIP_XCVR_CFG0_0

UTMIP transceiver cell configuration register 0. UTMIP REGISTER: UTMIP_XCVR_CFG0

Define spec-dependent variable

Offset: 808h | Read/Write: R/W | Reset: 0b00100000001001010110010100000000

Bit	Reset	Description
31:25	0x10	UTMIP_XCVR_HSSLEW_MSB: Most significant bits of HS_SLEW.
24:22	0x0	UTMIP_XCVR_SETUP_MSB: Most significant bits of SETUP.

Bit	Reset	Description
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.

26.5.6.10 USB3_UTMIP_BIAS_CFG0_0

UTMIP Bias cell configuration register 0

UTMIP REGISTER: UTMIP_BIAS_CFG0

Offset: 80ch | Read/Write: R/W | Reset: 0b00000000000000100000000000

Bit	Reset	Description
24	0x0	UTMIP_HSDISCON_LEVEL_MSB: Most significant bit of UTMIP_HSDISCON_LEVEL, bit 2
23	0x0	UTMIP_IDPD_VAL: See IDPD_SEL.
22	0x0	UTMIP_IDPD_SEL: 0: IDPD = ~IdPullup. 1: IDPD = IDPD_VAL.
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 circuit.
10	0x0	UTMIP_BIASPD: Power down bias circuit.

Bit	Reset	Description
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.

26.5.6.11 USB3_UTMIP_HSRX_CFG0_0

UTMIP High speed receive config 0

UTMIP REGISTER: UTMIP_HSRX_CFG0

Offset: 810h | Read/Write: R/W | Reset: 0b10010001011001010011010000000000

Bit	Reset	Description
31:30	0x2	UTMIP_KEEP_PATT_ON_ACTIVE: Keep the stay alive pattern on active
29	0x0	UTMIP_ALLOW_CONSEC_UPDN: Allow consecutive ups and downs on the bits, debug only, set to 0.
28	0x1	UTMIP_REALIGN_ON_NEW_PKT: Realign the inertia counters on a new packet
27:24	0x1	UTMIP_PCOUNT_UPDN_DIV: The number of (edges-1) needed to move the sampling point
23:21	0x3	UTMIP_SQUELCH_EOP_DLY: Limit the delay of the squelch at EOP time
20	0x0	UTMIP_NO_STRIPPING: Do not strip incoming data
19:15	0xa	UTMIP_IDLE_WAIT: Number of cycles of idle to declare IDLE.
14:10	0xd	UTMIP_ELASTIC_LIMIT: Depth of elastic input store
9	0x0	UTMIP_ELASTIC_OVERRUN_DISABLE: Do not declare overrun errors until overflow of FIFO
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

26.5.6.12 USB3_UTMIP_HSRX_CFG1_0

UTMIP High speed receive config 1

UTMIP REGISTER: UTMIP_HSRX_CFG1

Offset: 814h | Read/Write: R/W | Reset: 0b010011

Bit	Reset	Description
5:1	0x9	UTMIP_HS_SYNC_START_DLY: How long to wait before start of sync launches RxActive
0	0x1	UTMIP_HS_ALLOW_KEEP_ALIVE: Allow Keep Alive packets

26.5.6.13 USB3_UTMIP_FSLSRX_CFG0_0

UTMIP full and Low speed receive config 0

UTMIP REGISTER: UTMIP_FSLSRX_CFG0

Offset: 818h | Read/Write: R/W | Reset: 0b11111101010101001000010000101001

Bit	Reset	Description
31	0x1	UTMIP_FSLS_SE1_DRIBBLE_FILTER: One SE1, don't allow dribble
30	0x1	UTMIP_FSLS_SE1_FILTER: Filter SE1
29	0x1	UTMIP_FSLS_SERIAL_SE0_RCV
28:26	0x7	UTMIP_FSLS_UPR_DRIBBLE_SIZE: Do not allow <= dribble bits
25:23	0x2	UTMIP_FSLS_LWR_DRIBBLE_SIZE: Do not allow >= dribble bits
22	0x1	UTMIP_FSLS_EOP_ENDS_AT_SE0: Only look for transitioning out of EOP
21:16	0x14	UTMIP_FSLS_KCOUNT_MAX: Number of K bits in question
15	0x1	UTMIP_FSLS_KCOUNT_LIMIT: Limit the number of bit times a K can last
14	0x0	UTMIP_FSLS_ACTIVE_ON_FULL_SYNC: Require a full sync pattern to declare the data received
13:8	0x4	UTMIP_FSLS_IDLE_WAIT_MAX: 4 bits of of SEO should exceed the time limit
7	0x0	UTMIP_FSLS_IDLE_WAIT_LIMIT: Enable the reset of the state machine on extended SE0
6:1	0x14	UTMIP_FSLS_IDLE_COUNT_MAX: 20 bits of idle should end the packet if FsLsIdleCountLimitCfg=1.
0	0x1	UTMIP_FSLS_IDLE_COUNT_LIMIT: Give up on packet if a long sequence of J

26.5.6.14 USB3_UTMIP_FSLSRX_CFG1_0

UTMIP full and Low speed receive config 1

Offset: 81ch | Read/Write: R/W | Reset: 0b01000100110011110100000000000

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

Bit	Reset	Description
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

26.5.6.15 USB3_UTMIP_TX_CFG0_0

UTMIP transmit config signals

UTMIP REGISTER: UTMIP_TX_CFG0

Offset: 820h | Read/Write: R/W | Reset: 0b00010000001000000000

Bit	Reset	Description
19	0x0	UTMIP_FS_PREAMBLE_J: output enable sends an initial J before sync pattern
18	0x0	UTMIP_FS_POSTAMBLE_OUTPUT_ENABLE: output enable turns off 1/2 cycle after
17	0x0	UTMIP_FS_PREAMBLE_OUTPUT_ENABLE: output enable turns on 1/2 cycle before
16	0x1	UTMIP_FSLS_ALLOW_SOP_TX_STUFF_ERR: Allow SOP to be source of transmit error stuffing
15	0x0	UTMIP_HS_READY_WAIT_FOR_VALID
14:10	0x0	UTMIP_HS_TX_IPG_DLY
9	0x1	UTMIP_HS_DISCON_EOP_ONLY: Only check during EOP
8	0x0	UTMIP_HS_DISCON_DISABLE: Disable high speed disconnect
7	0x0	UTMIP_HS_POSTAMBLE_OUTPUT_ENABLE: output enable turns off 1 cycle after
6	0x0	UTMIP_HS_PREAMBLE_OUTPUT_ENABLE: output enable turns on 1 cycle before
5	0x0	UTMIP_SIE_RESUME_ON_LINESTATE: SIE, not macrocell, detects LineState change to resume
4	0x0	UTMIP_SOF_ON_NO_STUFF: Sof when OpMode 3 -- perhaps, when sending controller made packets
3	0x0	UTMIP_SOF_ON_NO_ENCODE: Sof when OpMode 2 -- not likely, for Chirp
2	0x0	UTMIP_NO_STUFFING: No bit stuffing, static programming
1	0x0	UTMIP_NO_ENCODING: No encoding, static programming
0	0x0	UTMIP_NO_SYNC_NO_EOP: Do not sent SYNC or EOP

26.5.6.16 USB3_UTMIP_MISC_CFG0_0

UTMIP miscellaneous configurations

UTMIP REGISTER: UTMIP_MISC_CFG0

Offset: 824h | Read/Write: R/W | Reset: 0b00000111110000000000000001111000

Bit	Reset	Description
30:27	0x0	UTMIP_DPDM_OBSERVE_SEL: Select DP/DM obs signals
26	0x0	UTMIP_DPDM_OBSERVE: Use DP/DM as obs bus
25	0x1	UTMIP_KEEP_XCVR_PD_ON_SOFT_DISCON
24	0x1	UTMIP_ALLOW_LS_ON_SOFT_DISCON
23	0x1	UTMIP_FORCE_FS_DISABLE_ON_DEV_CHIRP
22	0x1	UTMIP_SUSPEND_EXIT_ON_EDGE: Suspend exit requires edge or simply a value...
21	0x1	UTMIP_LS_TO_FS_SKIP_4MS: Don't block changes for 4ms when going from LS to FS (should not happen)
20:19	0x0	UTMIP_INJECT_ERROR_TYPE: Force error insertion into RX path. (Used for IOBIST.) 0 = DISABLE 1 = BIT_ERR 2 = RX_ERR 3 = BIT_RX_ERR
18	0x0	UTMIP_FORCE_HS_CLOCK_ON: Force HS clock always on.
17	0x0	UTMIP_DISABLE_HS_TERM: Force HS termination inactive.
16	0x0	UTMIP_FORCE_HS_TERM: Force HS termination active.
15	0x0	UTMIP_DISABLE_PULLUP_DP: Force DP pullup inactive. (Overrides FORCE_PULLUP_DP.)
14	0x0	UTMIP_DISABLE_PULLUP_DM: Force DM pullup inactive. (Overrides FORCE_PULLUP_DM.)
13	0x0	UTMIP_DISABLE_PULLDN_DP: Force DP pulldown inactive. (Overrides FORCE_PULLDN_DP.)
12	0x0	UTMIP_DISABLE_PULLDN_DM: Force DM pulldown inactive. (Overrides FORCE_PULLDN_DM.)
11	0x0	UTMIP_FORCE_PULLUP_DP: Force DP pullup active.
10	0x0	UTMIP_FORCE_PULLUP_DM: Force DM pullup active.
9	0x0	UTMIP_FORCE_PULLDN_DP: Force DP pulldown active.
8	0x0	UTMIP_FORCE_PULLDN_DM: Force DM pulldown active.
7:5	0x3	UTMIP_STABLE_COUNT: Number of cycles of crystal clock of signal not changing to consider stable.
4	0x1	UTMIP_STABLE_ALL: Determines if all signal need to be stable to not change a config.
3	0x1	UTMIP_NO_FREE_ON_SUSPEND: Don't use free running terminations during suspend.
2	0x0	UTMIP_NEVER_FREE_RUNNING_TERMS: Ignore free running terminations, even when no clock
1	0x0	UTMIP_ALWAYS_FREE_RUNNING_TERMS: Use free running terminations at all time
0	0x0	UTMIP_COMB_TERMS: Use combinational terminations or synced through CLKXTAL

26.5.6.17 USB3_UTMIP_MISC_CFG1_0

UTMIP miscellaneous configurations

UTMIP REGISTER: UTMIP_MISC_CFG1

Offset: 828h | Read/Write: R/W | Reset: 0b10000000001100110000000000100100

Bit	Reset	Description
30	0x1	UTMIP_PHY_XTAL_CLOCKEN: Selects whether to enable the crystal clock in the module.
29	0x0	UTMIP_LINESTATE_BYPASS: Bypass LineState reclocking logic
28	0x0	UTMIP_LINESTATE_NEG: Use neg edge sync for linestate
27	0x0	UTMIP_LINESTATE_XCVRSEL3: 0: Use FS filtering on line state when XcvrSel=3 1: Use LS filtering on line state when XcvrSel=3
26:25	0x0	UTMIP_OBS_SEL
24	0x0	UTMIP_FSLS_TDM
23	0x0	UTMIP_FORCE_IOBIST_CLK_ON
22:18	0x6	UTMIP_PLL_ACTIVE_DLY_COUNT: 5 us / (1/19.2MHz) = 96 / 16 = 6
17:6	0x600	UTMIP_PLLU_STABLE_COUNT: WRONG! This should be 1ms -> 0x50
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

26.5.6.18 USB3_UTMIP_DEBOUNCE_CFG0_0

UTMIP Avalid and Bvalid debounce

UTMIP REGISTER: UTMIP_DEBOUNCE_CFG0

Debounce values IdDig, Avalid, Bvalid, VbusValid, VbusWakeUp, and SessEnd. Each of these signals have their own debouncer and for each of those one out of 2 debouncing times can be chosen (BIAS_DEBOUNCE_A or BIAS_DEBOUNCE_B.)

The values of DEBOUNCE_A and DEBOUNCE_B are calculated as follows:

0xffff -> No debouncing at all

ms = *1000 / (1/19.2MHz) / 4

So to program a 1 ms debounce for BIAS_DEBOUNCE_A:

BIAS_DEBOUNCE_A[15:0] = 1000 * 19.2 / 4 = 4800 = 0x12c0

Offset: 82ch | Read/Write: R/W | Reset: 0b11111111111111111111111111111111

Bit	Reset	Description
31:16	0xffff	UTMIP_BIAS_DEBOUNCE_B: simulation value -- Used for interrupts

Bit	Reset	Description
15:0	0xffff	UTMIP_BIAS_DEBOUNCE_A: simulation value -- Used for interrupts

26.5.6.19 USB3_UTMIP_BAT_CHRG_CFG0_0

UTMIP battery charger configuration

UTMIP REGISTER: UTMIP_BAT_CHRG_CFG0

Offset: 830h | Read/Write: R/W | Reset: 0b00001

Bit	Reset	Description
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

26.5.6.20 USB3_UTMIP_SPARE_CFG0_0

Utmip spare configuration bits

UTMIP REGISTER: UTMIP_SPARE_CFG0

Offset: 834h | Read/Write: R/W | Reset: 0b11111111111111110000000000000000

Bit	Reset	Description
31:0	-65536	UTMIP_SPARE: Spare register bits 0: HS_RX_IPG_ERROR_ENABLE 1: HS_RX_FLUSH_ALAP. Flush as late as possible 2: HS_RX_LATE_SQUELCH. Delay Squelch by 1 CLK480 cycle. 31 to 3: Reserved

26.5.6.21 USB3_UTMIP_XCVR_CFG1_0

UTMIP transceiver cell configuration register 1

UTMIP REGISTER: UTMIP_XCVR_CFG1

Offset: 838h | Read/Write: R/W | Reset: 0b00000000010000010001111111

Bit	Reset	Description
23:22	0x0	UTMIP_XCVR_SPARE: Spare bits for usb transceiver pad ECO.
21:18	0x0	UTMIP_XCVR_TERM_RANGE_ADJ: Range adjusment 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	0x1	UTMIP_FORCE_PDDR_POWERUP: Force PDDR input into power up.
4	0x1	UTMIP_FORCE_PDDR_POWERDOWN: Force PDDR input input into power down. (Overrides

Bit	Reset	Description
		FORCE_PDDR_POWERUP.)
3	0x1	UTMIP_FORCE_PDCHRP_POWERUP: Force PDCHRP input into power up.
2	0x1	UTMIP_FORCE_PDCHRP_POWERDOWN: Force PDCHRP input input into power down. (Overrides FORCE_PDCHRP_POWERUP.)
1	0x1	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.)

26.5.6.22 USB3_UTMIP_BIAS_CFG1_0

UTMIP Bias cell configuration register 1

UTMIP REGISTER: UTMIP_BIAS_CFG1

Offset: 83ch | Read/Write: R/W | Reset: 0b00000000101010

Bit	Reset	Description
13:8	0x0	UTMIP_BIAS_DEBOUNCE_TIMESCALE: Debouncer time scaling, factor-1 to slow down debouncing by. So 0 is 1, 1 is 2, etc.
7:3	0x5	UTMIP_BIAS_PDTRK_COUNT: Control the BIAS cell power down lag. The lag should be 20us. For a Xtal clock of 13MHz it should be set a 5.
2	0x0	UTMIP_VBUS_WAKEUP_POWERDOWN: Force VBUS_WAKEUP input into power down.
1	0x1	UTMIP_FORCE_PDTRK_POWERUP: Force PDTRK input into power up.
0	0x0	UTMIP_FORCE_PDTRK_POWERDOWN: Force PDTRK input into power down. (Overrides FORCE_PDTRK_POWERUP.)

26.5.6.23 USB3_UTMIP_BIAS_STS0_0

UTMIP Bias cell status register 0

UTMIP REGISTER: UTMIP_BIAS_STS0

Offset: 840h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	UTMIP_TCTRL: Thermal encoding output from USB bias pad.
15:0	X	UTMIP_RCTRL: Thermal encoding output from USB bias pad.

26.5.6.24 USB2_QH_USB2D_QH_EP_0_OUT_0

USB2D Queue Head for OUT endpoint 0

Offset: 1000h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 0 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 0.

26.5.6.25 USB2_QH_USB2D_QH_EP_0_IN_0

USB2D Queue Head for IN endpoint 0

Offset: 1040h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 0. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 0.

26.5.6.26 USB2_QH_USB2D_QH_EP_1_OUT_0

USB2D Queue Head for OUT endpoint 1

Offset: 1080h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 1 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 1.

26.5.6.27 USB2_QH_USB2D_QH_EP_1_IN_0

USB2D Queue Head for IN endpoint 1

Offset: 10c0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 1. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 1.

26.5.6.28 USB2_QH_USB2D_QH_EP_2_OUT_0

USB2D Queue Head for OUT endpoint 2

Offset: 1100h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 2 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 2.

26.5.6.29 USB2_QH_USB2D_QH_EP_2_IN_0

USB2D Queue Head for IN endpoint 2

Offset: 1140h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 2. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 2.

26.5.6.30 USB2_QH_USB2D_QH_EP_3_OUT_0

USB2D Queue Head for OUT endpoint 3

Offset: 1180h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 3 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 3.

26.5.6.31 USB2_QH_USB2D_QH_EP_3_IN_0

USB2D Queue Head for IN endpoint 3

Offset: 11c0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 3. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 3.

26.5.6.32 USB2_QH_USB2D_QH_EP_4_OUT_0

USB2D Queue Head for OUT endpoint 4

Offset: 1200h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 4 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 4.

26.5.6.33 USB2_QH_USB2D_QH_EP_4_IN_0

USB2D Queue Head for IN endpoint 4

Offset: 1240h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 4. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 4.

26.5.6.34 USB2_QH_USB2D_QH_EP_5_OUT_0

USB2D Queue Head for OUT endpoint 5

Offset: 1280h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 5 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 5.

26.5.6.35 USB2_QH_USB2D_QH_EP_5_IN_0

USB2D Queue Head for IN endpoint 5

Offset: 12c0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 5. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 5.

26.5.6.36 USB2_QH_USB2D_QH_EP_6_OUT_0

USB2D Queue Head for OUT endpoint 6

Offset: 1300h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 6 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 6.

26.5.6.37 USB2_QH_USB2D_QH_EP_6_IN_0

USB2D Queue Head for IN endpoint 6

Offset: 1340h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 6. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 6.

26.5.6.38 USB2_QH_USB2D_QH_EP_7_OUT_0

USB2D Queue Head for OUT endpoint 7

Offset: 1380h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 7 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 7.

26.5.6.39 USB2_QH_USB2D_QH_EP_7_IN_0

USB2D Queue Head for IN endpoint 7

Offset: 13c0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 7. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 7.

26.5.6.40 USB2_QH_USB2D_QH_EP_8_OUT_0

USB2D Queue Head for OUT endpoint 8

Offset: 1400h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 8 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 8.

26.5.6.41 USB2_QH_USB2D_QH_EP_8_IN_0

USB2D Queue Head for IN endpoint 8

Offset: 1440h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 8. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 8.

26.5.6.42 USB2_QH_USB2D_QH_EP_9_OUT_0

USB2D Queue Head for OUT endpoint 9

Offset: 1480h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 9 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 9.

26.5.6.43 USB2_QH_USB2D_QH_EP_9_IN_0

USB2D Queue Head for IN endpoint 9

Offset: 14c0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 9. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 9.

26.5.6.44 USB2_QH_USB2D_QH_EP_10_OUT_0

USB2D Queue Head for OUT endpoint 10

Offset: 1500h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 10 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 10.

26.5.6.45 USB2_QH_USB2D_QH_EP_10_IN_0

USB2D Queue Head for IN endpoint 10

Offset: 1540h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 10. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 10.

26.5.6.46 USB2_QH_USB2D_QH_EP_11_OUT_0

USB2D Queue Head for OUT endpoint 11

Offset: 1580h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 11 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 11.

26.5.6.47 USB2_QH_USB2D_QH_EP_11_IN_0

USB2D Queue Head for IN endpoint 11

Offset: 15c0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 11. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 11.

26.5.6.48 USB2_QH_USB2D_QH_EP_12_OUT_0

USB2D Queue Head for OUT endpoint 12

Offset: 1600h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 12 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 12.

26.5.6.49 USB2_QH_USB2D_QH_EP_12_IN_0

USB2D Queue Head for IN endpoint 12

Offset: 1640h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 12. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 12.

26.5.6.50 USB2_QH_USB2D_QH_EP_13_OUT_0

USB2D Queue Head for OUT endpoint 13

Offset: 1680h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 13 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 13.

26.5.6.51 USB2_QH_USB2D_QH_EP_13_IN_0

USB2D Queue Head for IN endpoint 13

Offset: 16c0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 13. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 13.

26.5.6.52 USB2_QH_USB2D_QH_EP_14_OUT_0

USB2D Queue Head for OUT endpoint 14

Offset: 1700h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 14 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 14.

26.5.6.53 USB2_QH_USB2D_QH_EP_14_IN_0

USB2D Queue Head for IN endpoint 14

Offset: 1740h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 14. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 14.

26.5.6.54 USB2_QH_USB2D_QH_EP_15_OUT_0

USB2D Queue Head for OUT endpoint 15

Offset: 1780h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 15 This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 15.

26.5.6.55 USB2_QH_USB2D_QH_EP_15_IN_0

USB2D Queue Head for IN endpoint 15

Offset: 17c0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 15. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 15.

27.0 AUDIO SUBSYSTEM

27.1 Digital Audio Switch

The Digital Audio Switch (DAS) is a generic matrix which connects the three digital audio-controllers (DAC) in Tegra® 2 Series devices (I2S1, I2S2 and AC-97) to the five audio-ports (DAP) of the Tegra 2 Series chip. DAS uses seven sets of configuration-inputs for this purpose:

- DAP selects
- DAC clock input-selects
- DAC sdata1 selects
- DAC sdata2 selects
- DAP Bypass-mode Master-selects
- DAP bypass-mode sdata1 Tx Rx selects
- DAP bypass-mode sdata2 Tx Rx selects

These inputs are mapped to register-writes under misc_regs (APB_MISC_DAS_DAP_CTRL_SEL_0 and APB_MISC_DAS_DAC_INPUT_DATA_CLK_SEL_0).

Programmable connectivity for the DACs to any of the five available DAPs is a very useful feature. Any compatible codec connected on any one of the ports could be communicated to, by any of the three audio controllers inside the chip, without any change in the associated hardware. Also, bypass mode allows any of the DAPs to be connected to any other DAP either in master-mode or in slave-mode.

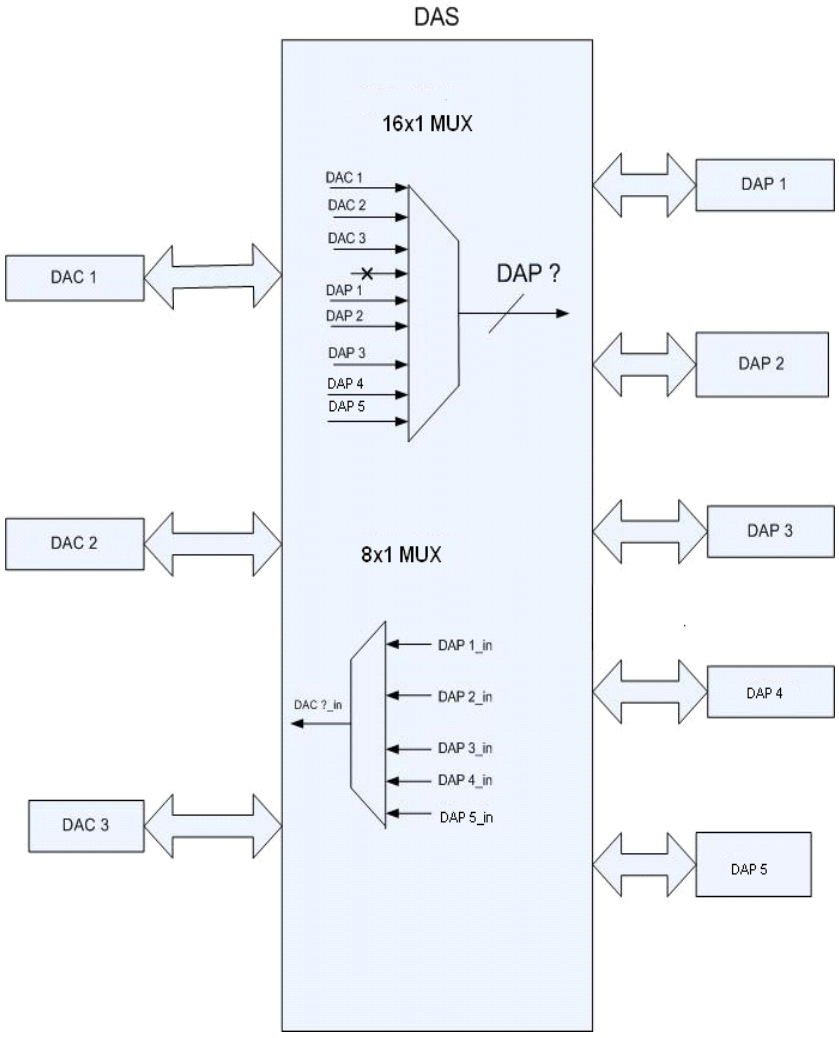
Features

The programmability would require three sets of configuration inputs:

- DAP-connectivity selects
- DAC input selects (clock, sdata1 and sdata2) and
- DAP master-selects in bypass modes

With these mux-selects, the DAS design comprises of instantiation of forty 16x1 muxes for the DAP-output connections, and twelve 8x1 muxes for the DAC input connections.

Figure 70. DAS with DACs and DAPs



27.1.1 Key Use Cases

- **DAC1 to all DAPs:** This verifies connectivity between DAC1 (I2S1) and all DAPs (DAP1-DAP5) individually. It covers both master and slave modes of connectivity, does a simultaneous record and playback (both data lines active).
- **DAC2 to all DAPs:** This verifies connectivity between DAC2 (I2S2) and all DAPs (DAP1-DAP5) individually. It covers both master and slave modes of connectivity, does a simultaneous record and playback (both data lines active).
- **DAC3 to all DAPs:** This verifies connectivity between DAC3 (AC97) and all DAPs (DAP1-DAP5) individually. It covers simultaneous record and playback (both data lines active).
- **DAC1 to all DAPs simultaneously:** This verifies connectivity between DAC1 (I2S1) and all DAPs (DAP1-DAP5) simultaneously. DAC1 can record data from only one of the four DAPs at any given point and so, the test covers selection of all possible sources for sdata_in from DAP1 to DAP5, one after the other.
- **DAC2 to all DAPs simultaneously:** This verifies connectivity between DAC2 (I2S2) and all DAPs (DAP1-DAP5) simultaneously. DAC2 can record data from only one of the four DAPs at any given point and so, the test covers selection of all possible sources for sdata_in from DAP1 to DAP5, one after the other
- **DAP to DAP bypass mode:** This verifies the bypass connectivity between DAPs (DAP1-DAP2, DAP2-DAP3, and so on). Both directions of data-transfer are tested with one of the DAPs in Master mode and the same is repeated with the other DAP in Master-mode.

- **All DAPs in bypass mode:** All DAPs are configured in Bypass mode, with one of the DAPs in Master mode and other four in Slave mode. Master-DAP transmits to the other four DAPs and can receive from only one of the other four DAPs.
- **DAC1 to DAP1 and DAP2-DAP1 bypass:** This is a use-case where the DAP2 is in by-pass connection with DAP1 and DAP1 is connected to DAC1. DAP1 is in Master mode and transmits to both DAC1 and DAP2. DAP1 receives data from only DAC1.
- **All DACs simultaneous:** This verifies the parallel operation of the three DACs through three DAPs, simultaneously.

27.1.2 Restrictions

- DAC-to-DAC loopback is not supported.
- When more than one DAP is configured to communicate to one DAC (in master-mode), the DAC can take inputs from only one of the DAPs. That is, it can transmit data to more than one DAP but can receive data from only one.
- When a DAC is in slave mode, and one of the DAPs is configured to be connected to it, no other DAP can be configured to connect to the same DAC to transmit or receive data.
- When a DAP is configured to connect to a given DAC, with DAC in master mode, no other DAP can connect meaningfully to this DAP in by-pass mode.
- When in Bypass mode, firmware needs to make sure it programs only one of the DAPs to be in master-mode.

27.1.3 Programming Guidelines

- DAC-DAP connectivity (Normal mode):
  - Program the APB_MISC_DAS_DAP_CTRL_SEL_? bit-fields as desired. (Only the options 000, 001, 010 signify normal mode)
  - Program APB_MISC_DAS_DAC_INPUT_DATA_CLK_SEL_? to suitably select the source for the inputs to DAC (Select based on the APB_MISC_DAS_DAP_CTRL_SEL_? above)
  - For any DAC-DAP configuration, the corresponding DAP?_MS_SEL has no significance. Master-Slave configuration is controlled by the DAC and FW.
- DAP-DAP connectivity (Bypass mode):
  - Program the APB_MISC_DAS_DAP_CTRL_SEL_? bit-fields as desired. (Only the options 10000, 10001, 10010, 10011 and 10100 signify Bypass mode)
  - Select DAP?M_SEL to define the Master-Slave relationship

27.1.3.1 Example Use-case Configuration

DAC1 to DAP1 and DAP2-DAP1 bypass: This is a use-case where the DAP2 is in by-pass connection with DAP1 and DAP1 is connected to DAC1. DAP1 is in Master mode and transmits to both DAC1 and DAP2. DAP1 receives data from only DAC1. DAP3 and DAP4 are connected to DAC2 (Master). DAP2 inputs are sourced only from DAP4. Following is the configuration is to be used:

- APB_MISC_DAS_DAP_CTRL_SEL_0[4:0] = 00000 (DAP1-DAC1)
- APB_MISC_DAS_DAP_CTRL_SEL_1[4:0] = 10000 (Bypass to DAP1)
- DAP_MS_SEL in APB_MISC_DAS_DAP_CTRL_SEL_0[31]: = 1 (DAP1 in Master: this is required as DAP2 is in Bypass mode)
- DAP_MS_SEL in APB_MISC_DAS_DAP_CTRL_SEL_1[31]: 0 (DAP2 in Slave: this is required as DAP2 is in Bypass mode)
- APB_MISC_DAS_DAC_INPUT_DATA_CLK_SEL_0[31:0] = 32'h0 (DAC1 will receive data from the Master DAP1)
- APB_MISC_DAS_DAP_CTRL_SEL_2 = 00001 (DAP3-DAC2)

- APB_MISC_DAS_DAP_CTRL_SEL_3 = 00001 (DAP4-DAC2)
- DAP_MS_SEL in APB_MISC_DAS_DAP_CTRL_SEL_2[31]: 0 (Can leave at its default value: not of any significance here)
- DAP_MS_SEL in APB_MISC_DAS_DAP_CTRL_SEL_3[31]: 0 (Can leave at its default value: not of any significance here)
- APB_MISC_DAS_DAC_INPUT_DATA_CLK_SEL_1[31:0] = 0x33000003 (DAC2 inputs sourced from DAP4- DAC2 can transmit to DAP3 and DAP4 but can receive from DAP4 only)

27.1.4 DAS Registers

27.1.4.1 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: c00h..c13h | Read/Write: R/W | Reset: 0b000xxxxxxxxxxxxxxxxxxxxxxx00000

Bit	Reset	Description
31	0x0	DAP_MS_SEL: This bit is programmed to put particular DAP is either in master or slave mode when two or more DAPs are in by-pass mode. 0 = SLAVE 1 = MASTER
30	0x0	DAP_SDATA1_TX_RX: To program sdata1 in either tx or rx mode when two or more DAPs are in by-pass mode 0 = TX 1 = RX
29	0x0	DAP_SDATA2_RX_TX: To program sdata2 in either tx or rx mode when two or more DAPs are in by-pass 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 |

27.1.4.2 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: c40h..c4bh | Read/Write: R/W | Reset: 0b00000000xxxxxxxxxxxxxxxxxxxx0000

Bit	Reset	Description
31:28	0x0	DAC_SDATA2_SEL: These bits are to control the selection of sdata2 input for DACs. 0 = DAP1 1 = DAP2 2 = DAP3 3 = DAP4 4 = DAP5
27:24	0x0	DAC_SDATA1_SEL: These bits are to control the selection of sdata1 input for DACs. 0 = DAP1 1 = DAP2 2 = DAP3 3 = DAP4 4 = DAP5
3:0	0x0	DAC_CLK_SEL: These bits are to control the selection of bit clock and fsync for DACs. 0 = DAP1 1 = DAP2 2 = DAP3 3 = DAP4 4 = DAP5

27.2 I2S Controller

The I2S Controller streams synchronous serial audio data between system memory and an external audio device. The controller supports the I2S Left Justified Mode, Right Justified Mode, and DSP mode formats.

The I2S Controller also supports the PCM, telephony (network) and Time Divisioned Multiplex (TDM) modes of data-transfer. Pulse-Code-Modulation (PCM) is a standard method used to digitize audio (particularly voice) patterns for transmission over digital communication channels. The Telephony or Network mode is used to transmit and receive data to/from an external mono codec in a slot-based scheme of time-division multiplexing. TDM mode is same as network mode, but in TDM mode, all or any of the slots are active. All of these modes are synced up by a frame-sync signal, much the same as the left-right-channel-clock LRCK (in the case of normal I2S communication).

This controller is hooked onto the APB bus and can trigger the APB-bridge DMA to transfer data to and from its FIFOs. Controller supports two-FIFOs, FIFO-1 and FIFO-2 which can operate bi-directional in normal I2S modes but only unidirectional in PCM/Network/TDM modes.

Features

- PCM, Network and TDM mode support
- Can operate in both Master and Slave modes
- Supports I2S, RJM, LJM and DSP mode data-formats
- Supports non- 50:50 mark-space ratio in both Master and Slave modes of I2S, RJM, and LJM
- Two FIFOs and control logic. In I2S basic modes, both the FIFOs can be configured for Tx or Rx. In PCM/Network mode, FIFO-1 acts as Tx-FIFO and FIFO-2 as Rx-FIFO
- Buffer memory optimized for peripheral bus utilization, allowing multiple sets of data being stored for both incoming and outgoing data stream per slot (Data-Packed mode in both FIFOs)
- Supports PCM-mode mono and Network-mode with independent slot-selection for Tx-and-Rx
- PCM mode supports short and long FSYNC. Long FSYNC can be only 2-clocks wide in Master mode and any number of clocks in slave-mode
- Same LRCK/FSYNC for both Transmission and reception of data
- Network mode with independent slot-selection for both Tx and Rx
- TDM mode with flexibility in number of slots and slot(s) selection
- Capability to drive out a High-Z outside the prescribed slot for transmission
- 6-bit clock divisor to support single- and double-speed for the following rates: 8 KHz, 32 KHz, 44.1 KHz, 48 KHz, 96 KHz, and 128 KHz
- Maximum frequency of device clock supported is 24 MHz
- The number of slots can be programmed from 1 to maximum 8 in TDM mode.

The I2S controller supports bidirectional audio streams. It can operate in half-duplex or full-duplex mode; that is, it can transmit or receive data on both the I2S_DIN and I2S_DOUT pins, based on the register-configuration. The I2S controller supports input word lengths of 16, 20, 24, or 32 bits. In TDM mode, the word length can be supported in multiple of 4 starts from 8 bit.

In Master mode, the I2S controller signals the synchronization of all I2S data transactions. In Master mode, the I2S controller module generates the SCLK bit clock (I2S_SCLK) and Left-Right channel clock LRCK (I2S_LRC) to the A/D or D/A converter. The A/D or D/A converter synchronizes the input or output audio data based on these clock signals.

In Slave mode, the external A/D or D/A converter provides the synchronization clocks, bit clock (I2S_SCLK) and Left-Right channel clock (I2S_LRCK) to the I2S controller. The I2S controller synchronizes the input/output audio data to these clocks.

27.2.1 External Signaling

I2S interface consists of 4 signals--FSYNC, BITCLK, SDATA1 and SDATA2-- all defined as bidirectional. SDATA1 and SDATA2 associate themselves to FIFO-1 and FIFO-2 respectively. In I2S mode, the SDATA1 and SDATA2 can be configured to be either input or output based on the configuration of the corresponding FIFO.

27.2.1.1 Data Formats in Basic I2S Mode

Figure 71. I2S Mode

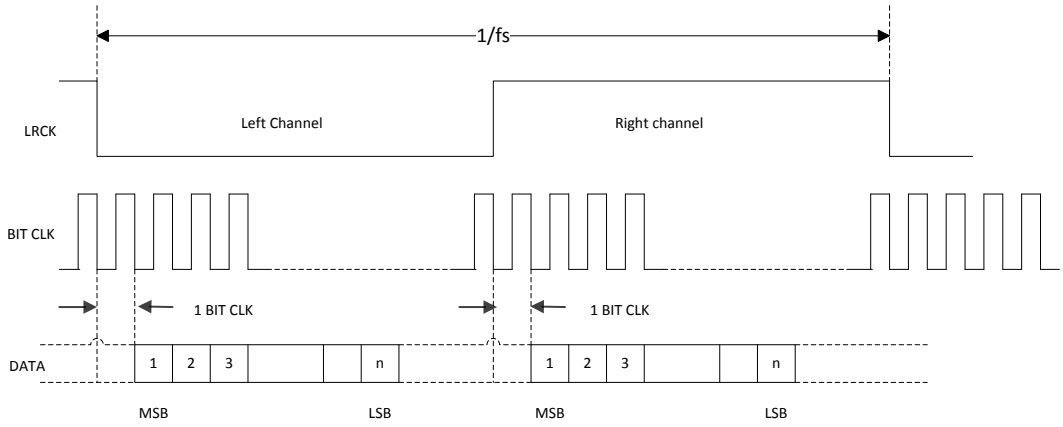


Figure 72. RJ Mode

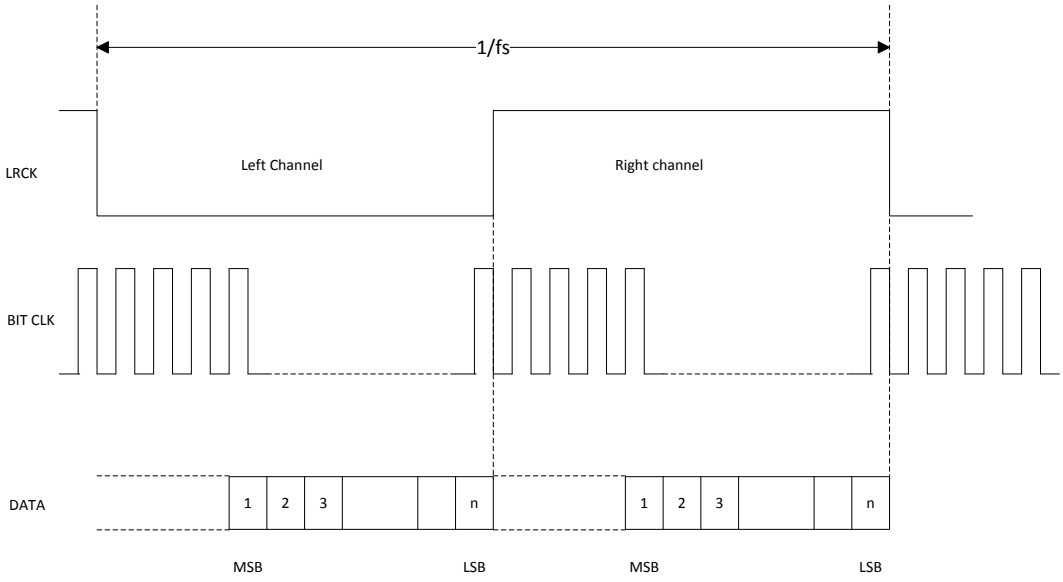


Figure 73. LJ Mode

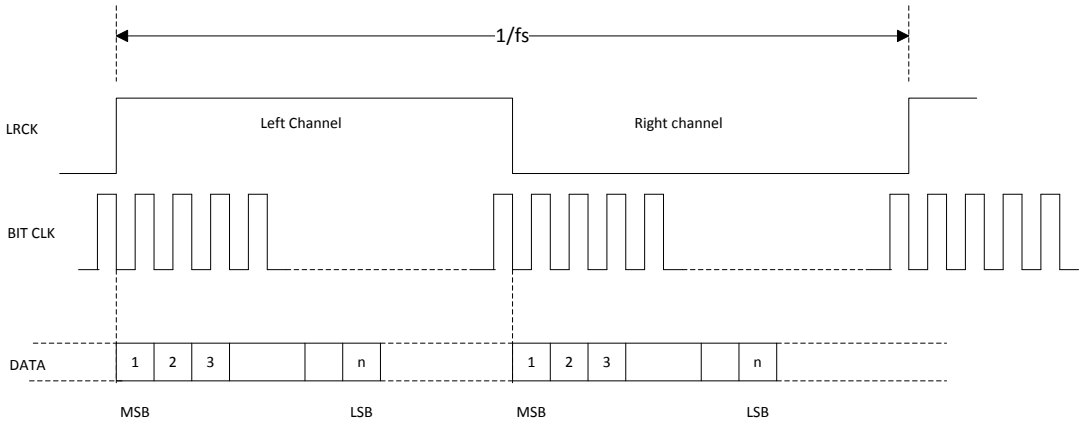
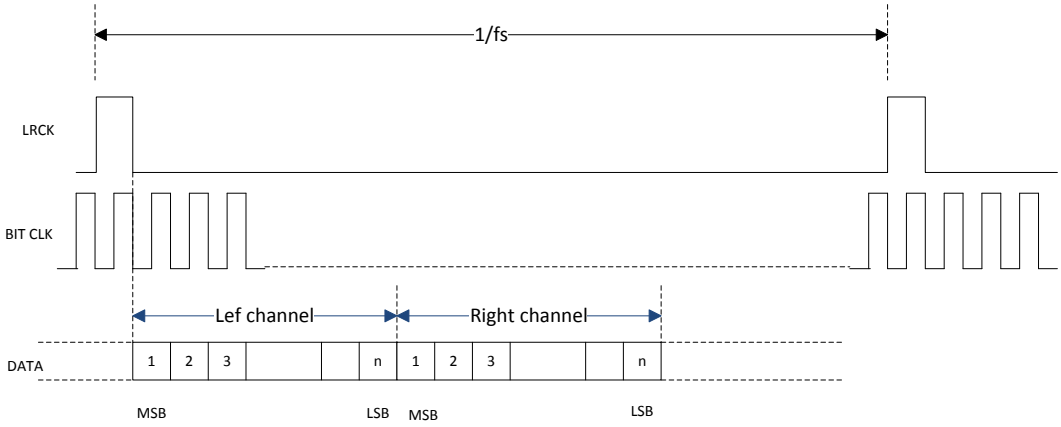
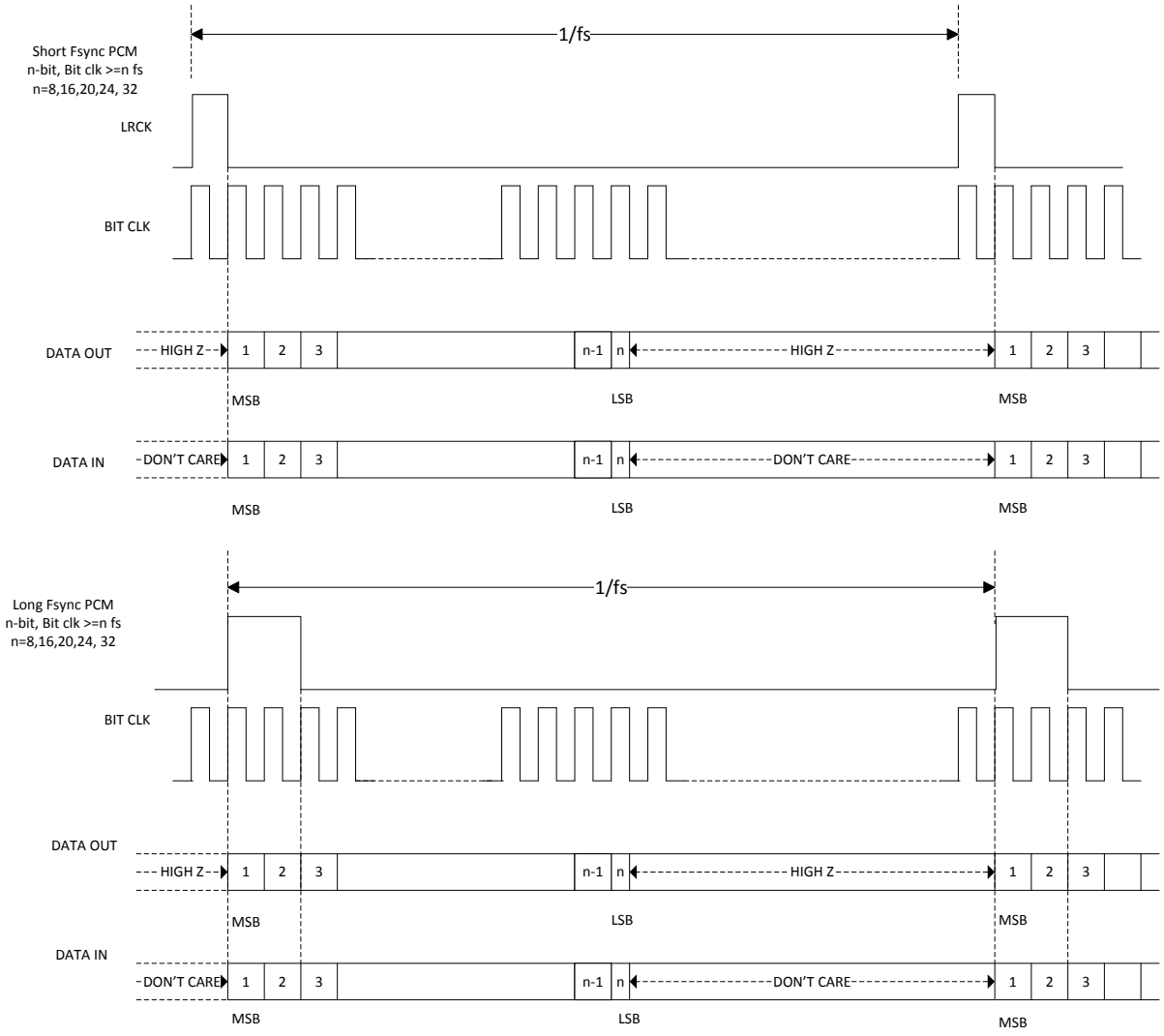


Figure 74. DSP Mode



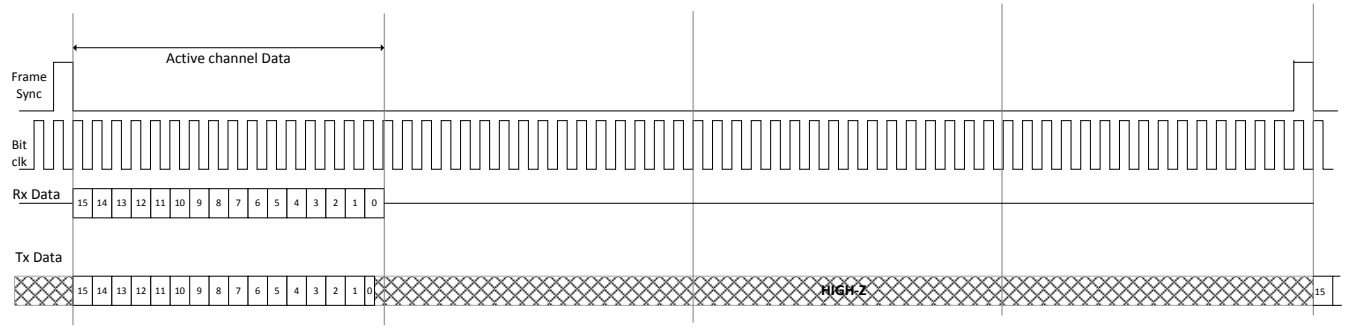
27.2.1.2 Data Formats in PCM Mode

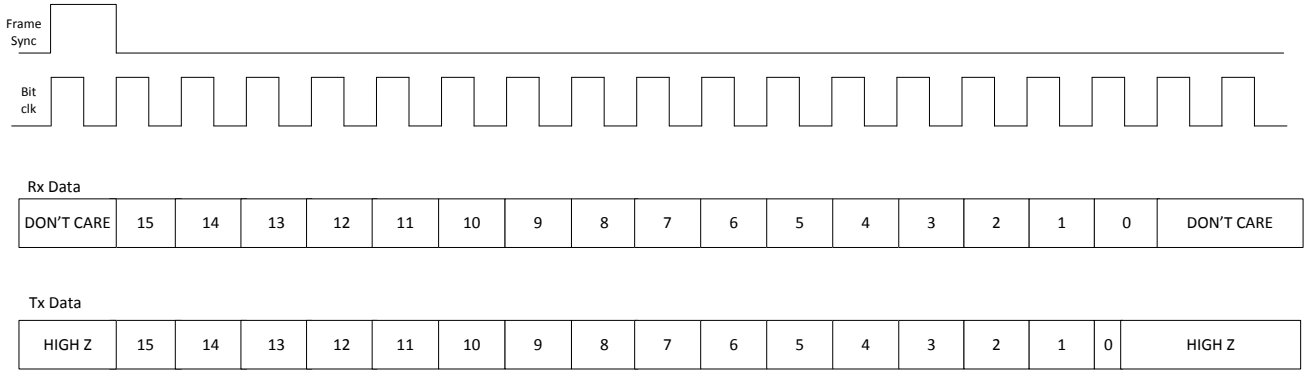
Figure 75. PCM Mode



27.2.1.3 Data Formats in Network Mode

Figure 76. Network Mode





27.2.1.4 Data format in TDM Mode

Figure 77. TDM Mode

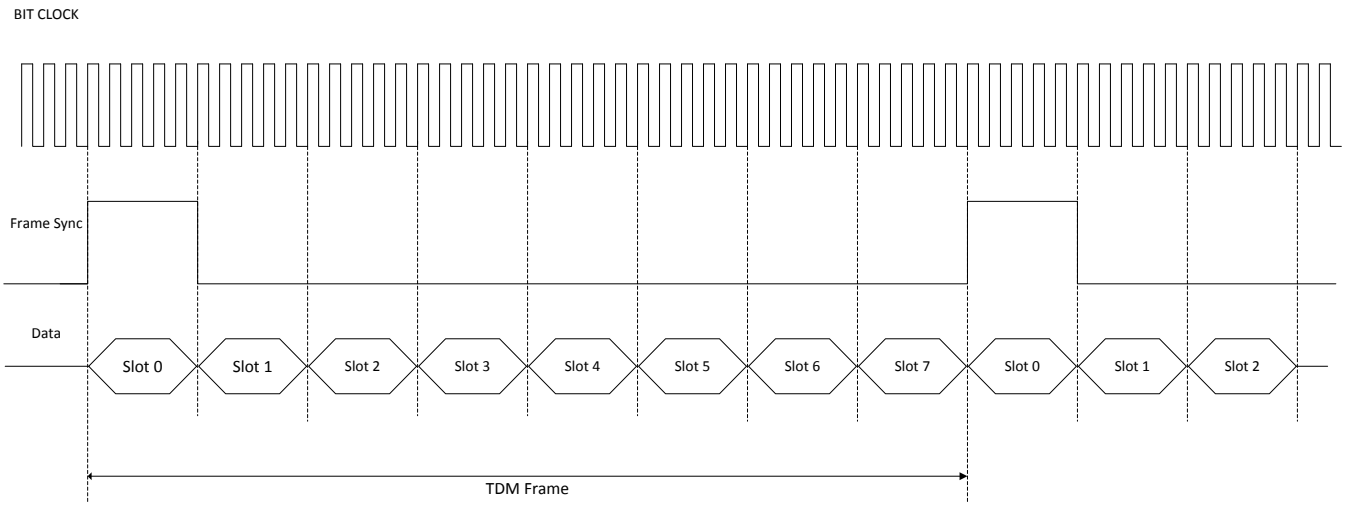
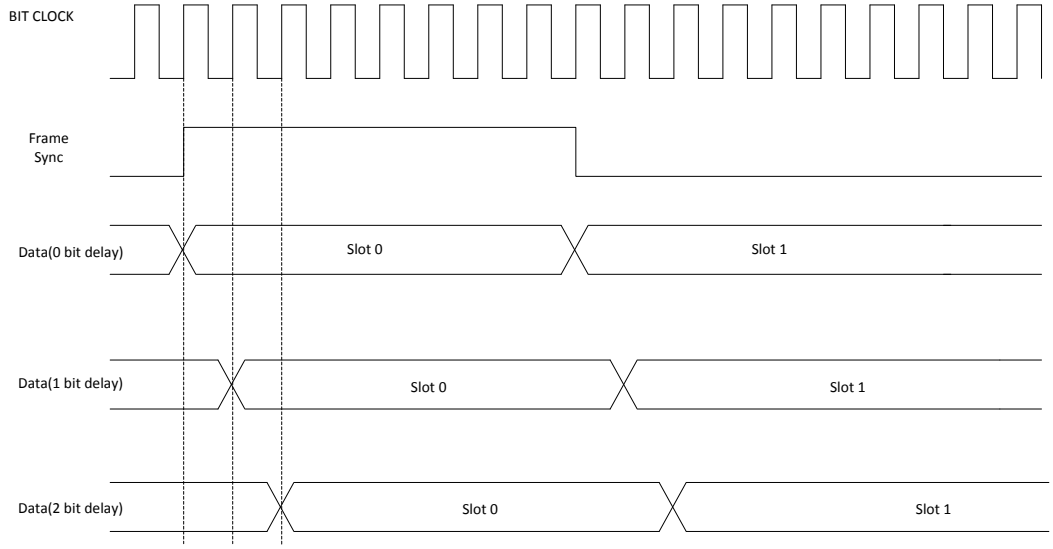
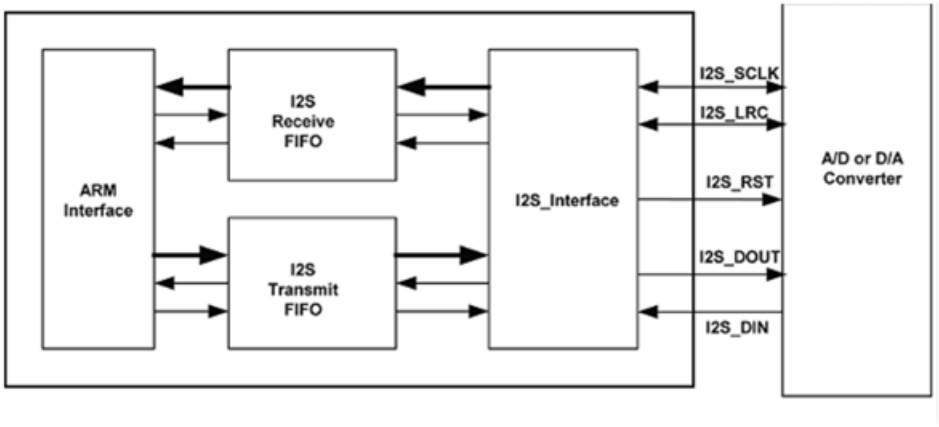


Figure 78. TDM Sync



27.2.2 Functionality

Figure 79. I2S Functional Block Diagram



27.2.3 Programming Guidelines

27.2.3.1 Module Initialization

Reset the controller (not mandatory after a POR) and enable the clocks to the controller.

27.2.3.2 I2S Sampling Rate selection

The channel_bit-cnt can be calculated using the relation:

```
Channel_bit_cnt = (frequency of bit_clk) / (2 * required sampling rate) -1;
```

If this calculation returns a fractional value, the non-symmetry feature of the controller should be used to attain the required sampling rate. In that case, the channel_bit_cnt value should be programmed with an integer that is closest to the fraction, but less than the fraction.

The table below shows some examples for common sampling rates, with CLK_SOURCE_I2S = 24 MHz

Table 86. Common Sampling Rates (CLK_Source_I2S = 24 MHz)

Sampling Rate	I2S_NEW_TIMING[12:00] (mark-space ratio:: Left_channel : Right_channel)				
	CLK_DIVISOR=0 BIT_CLK=24 MHz	CLK_DIVISOR=1 BIT_CLK=12 MHz	CLK_DIVISOR=3 BIT_CLK=6 MHz	CLK_DIVISOR=5 BIT_CLK=4 MHz	CLK_DIVISOR=7 BIT_CLK=3 MHz
8 KHz	0x05DB	0x02ED	0x0176	0x00F9	0x10BB
	(1500:1500)	(750:750)	(375:375)	(250:250)	(187:188)
32 KHz	0x0176	0x10BB	Not supported	0x103E	Not supported
	(375:375)	(187:188)		(62:63)	
44.1 KHz	0x010F	0x0087	0x0043	0x002C	0x0021
	(272:272)	(136:136)	(68:68)	(45:45)	(34:34)
48 KHz	0x00F9	0x007C	0x103E	Not supported	Not supported
	(250:250)	(125:125)	(62:63)		
96 KHz	0x007C	0x103E	0x101F	Not supported	Not supported
	(125:125)	(62:63)	(31:32)		

Note: Ideally, any sampling rate can be generated, either accurately or approximately with any clk_src selected for I2S, if the clk_divisor and the channel_bit_cnt are programmed accordingly.

NON_SYM.EN feature is meant to create the sampling rates approximately, by realizing an odd bit-rate with a non-50:50 mark/space ratio. However, if the bit-rate (2*channel_bit_cnt) itself is a fraction, the resultant sampling rate will not be accurate. The entries shown as not supported in the above table are attributed to such a deviation.

When the channel_bit_cnt is less than 32, the bit_size should be programmed to be less than channel_bit_cnt.

27.2.3.3 I2S Basic Transmit/Receive Data Flow

After the module-initialization (reset and clock programming), program the corresponding bits of the following registers, preferably in the same sequence:

1. **I2S_CTRL:** Set FIFOs to Tx/Rx-mode as required: set I2S_CTRL[27]=0/1 for FIFO1 to be in Tx/Rx-mode and I2S_CTRL[30]=1/0 for FIFO2 to be in Tx/Rx-mode.
2. **I2S_FIFO_SCR:** Clear the FIFOs before every new transaction and program the attention-levels.
3. **I2S_TIMING:** Program to match the required sampling-rate based on the clock-source chosen on i2s_div_clock.
4. **I2S_CTRL:** Program all bits except bits 29 and 28. Note that bits 27 and 30 are already programmed and they are not supposed to be over-written now.
5. **I2S_CTRL:** Enable transaction for the FIFOs as required (If transmitting data, the FIFO in Tx-mode should have some valid data before enabling the transaction on this FIFO. Tx-FIFO should never be allowed to underrun and Rx-FIFO should never be allowed to overrun.)
6. **I2S_STATUS:** Once the transaction is disabled, poll for the FIFO?_BSY to get de-asserted.

27.2.4 Network and PCM Modes

When in either Network or PCM mode, to get correct fsync, normal bit format should be in DSP mode along with control signals of Network and PCM control registers. In Network mode, channel bit count should always be equal or greater than 4 times the bit size, where bit size is the total number of bits to be transmitted or received in a particular slot.

- I2S_FIFO_SCR (Set FIFO1 to Tx-mode and FIFO2 to Rx-mode: set I2S_CTRL[27]=0 and I2S_CTRL[30]=0).
- I2S_FIFO_SCR (Clear the FIFOs before every new transaction and program the attention-levels)
- I2S_PCM_CNTRL/I2S_NW_CNTRL (Program all the bits except TRM_MODE and RCV_MODE)
- I2S_TIMING (Program to match the required sampling-rate based on the clock-source chosen on i2s_div_clock. In PCM-mode, channel-bit-count >= bit-size. In Network-mode, channel_bit_count >= 4*bit_size)
- I2S_CTRL (program all bits except bits 29 and 28: Note that bits 27 and 30 are already programmed and they are not supposed to be over-written now)
- I2S_PCM_CNTRL/I2S_NW_CNTRL (Assert TRM_MODE and/or RCV_MODE)
- I2S_CTRL: Enable transaction for the FIFOs as required (If Transmitting data, the FIFO in Tx-mode should have some valid data before enabling the transaction on this FIFO. Tx-FIFO should never be allowed to underrun and Rx-FIFO should never be allowed to overrun)
- I2S_STATUS: Once the transaction is disabled, poll for the FIFO?_BSY to get de-asserted.

27.2.5 TDM mode

To operate in TDM mode, I2S_TDM_CTRL[31:31] should be programmed to 1. In TDM mode, channel bit count should be equal or more than number of slots times bit size, where bit size is the total no of bits to be transmitted or received in a particular slot.

- I2S_FIFO_SCR (Clear the FIFOs before every new transaction and program the attention-levels)
- I2S_TDM_CNTRL/I2S_TDM_TX_RX_CNTRL (Program all the bits except TX_EN and RX_EN)
- I2S_TIMING (Program to match the required sampling-rate based on the clock-source chosen on i2s_div_clock. In TDM -mode, channel-bit-count >= No. of slots * bit-size.
- I2S_TDM_TX_RX_CTRL: Enable transaction for the FIFOs as required (If Transmitting data, the FIFO in Tx-mode should have some valid data before enabling the transaction on this FIFO. Tx-FIFO should never be allowed to underrun and Rx-FIFO should never be allowed to overrun)
- I2S_TDM_TX_RX_CNTRL: TX and Rx status. Once the transaction is disabled, poll for the ?X_BSY to get de-asserted.

27.2.6 I2S Registers

27.2.6.1 I2S_I2S_CTRL_0

This is I2S control register for bit formats (I2S, LJM, RJM or DSP), bit sizes (16, 20, 24 and 32), FIFO formats, Master/slave selection, LR polarity and error interrupt enables.

I2S Control Register

Offset: 000h | Read/Write: R/W | Reset: 0b0000000xxxxxxxxxxxx0000x0000000

Bit	Reset	Description
30	0x0	TX2_ENABLE: Configure FIFO-2 as Tx-FIFO 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
29	0x0	FIFO1_TRANSACTION_EN: Enable I2S Transaction on FIFO1 (Tx-enable if FIFO1 is in Tx mode and Rx-enable if FIFO1 is in Rx mode) 0 = DISABLE 1 = ENABLE
28	0x0	FIFO2_TRANSACTION_EN: Enable I2S Transaction on FIFO2 (Tx-enable if FIFO2 is in Tx mode and Rx-enable if FIFO2 is in Rx mode) 0 = DISABLE 1 = ENABLE
27	0x0	RX2_ENABLE: Configure FIFO-1 as Rx-FIFO 0 = DISABLE 1 = ENABLE
26	0x0	LPBK: Tx1-Rx1 Loop Back Test Enable (Works only in Tx1-Rx1 mode i.e. FIFO1 is in Tx mode, FiFO2 is in Rx mode) 0 = DISABLE 1 = ENABLE
25	0x0	M_S: Controller Master/Slave mode selection. Master always supplies bit clock and fsync(lrck) 0 = SLAVE 1 = MASTER
24	0x0	L_R: Left/Right Control Polarity. 0= Left channel when LRCK is low, Right channel when LRCK is high, 1= vice versa 0 = LRCK_LOW 1 = LRCK_HIGH
11:10	0x0	BIT_FORMAT: Controller Bit format 0 = I2S 1 = RJM 2 = LJM 3 = DSP
9:8	0x0	BIT_SIZE: Bit Size 0 = BIT_SIZE_16 1 = BIT_SIZE_20 2 = BIT_SIZE_24 3 = BIT_SIZE_32
6:4	0x0	FIFO_FORMAT: FIFO format 0 = BIT_SIZE_16_LSB 1 = BIT_SIZE_20_LSB 2 = BIT_SIZE_24_LSB 3 = BIT_SIZE_32 7 = PACKED
3	0x0	IE_FIFO1_ERR: Interrupt when FIFO1 error detected (Error is FIFO1-Tx-underrun if FIFO1 is in Tx mode, FIFO1-Rx-overrun if FIFO1 is in Rx mode) 0 = DISABLE 1 = ENABLE
2	0x0	IE_FIFO2_ERR: Interrupt when FIFO2 error detected (Error is FIFO2-Tx-underrun if FIFO2 is in Tx mode, FIFO2-Rx-overrun if FIFO2 is in Rx mode) 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1	0x0	QE_FIFO1: Interrupt when FIFO1 triggers (Trigger is a FIFO1-write-request if FIFO1 is in Tx mode, FIFO1-read-request if FIFO1 is in Rx mode) 0 = DISABLE 1 = ENABLE
0	0x0	QE_FIFO2: Interrupt when FIFO2 triggers (Trigger is a FIFO2-write-request if FIFO2 is in Tx mode, FIFO2-read-request if FIFO2 is in Rx mode) 0 = DISABLE 1 = ENABLE

27.2.6.2 I2S_I2S_STATUS_0

This is status register where the status of controller can be determined. It reflects if the controller is ready for data transmission, or if the controller is busy with TX/RX data transmission, or FIFO got any under run/over run errors, or FIFO reaches trigger level.

I2S Status Register

Offset: 004h | Read/Write: R/W | Reset: 0b0000xxxxxxxxxxxxxxxxxxxxxxxx0010

Bit	Reset	Description
31	0x0	FIFO1_RDY: FIFO1-Transaction is synchronized to LR clock 0 = NOT_READY 1 = READY
30	0x0	FIFO2_RDY: FIFO2-Transaction is synchronized to LR clock 0 = NOT_READY 1 = READY
29	0x0	FIFO1_BSY: FIFO1-shifter is busy transferring data 0 = NOT_BUSY 1 = BUSY
28	0x0	FIFO2_BSY: FIFO2-shifter is busy transferring data 0 = NOT_BUSY 1 = BUSY
3	0x0	IS_FIFO1_ERR: FIFo1-Error flag 0 = NO_ERR 1 = ERR
2	0x0	IS_FIFO2_ERR: FIFO2-Error flag 0 = NO_ERR 1 = ERR
1	0x1	QS_FIFO1: FIFO1-trigger-status 0 = NOT_TRIGGERED 1 = TRIGGERED
0	0x0	QS_FIFO2: FIFO2-trigger-status 0 = NOT_TRIGGERED 1 = TRIGGERED

27.2.6.3 I2S_I2S_TIMING_0

The CHANNEL_BIT_CNT field of this register is used to program the number of bit-clks per channel of LRCK (sampling rate) that the I2S-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.
- All other modes: - Number of bit clocks (sclk) within each individual channel width (LEFT or RIGHT).

The NON_SYM.EN can be used to program the I2S-Controller to output a non-50:50 mark-space ratio on to the I2S bus. When the NON_SYM.EN Bit[12] is enabled, the Controller sends out exactly the programmed number of clock cycles on the left channel and one bit-clk greater on the right channel.

When bit-format is DSP, the channel_bit_cnt should be programmed as the required number of bit_clks in left channel + that in the right channel.

Program this register before enabling any of the bits 29/28 of I2S_CTRL register.

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 channel_bit-cnt can be calculated using the relation:

```
Channel_bit_cnt  = (frequency of bit_clk) / (2 * required sampling rate) -1;
```

If this calculation returns a fractional value, the non-symmetry feature of the controller should be used to attain the required sampling rate.

I2S Timing Register

Offset: 008h | Read/Write: R/W | Reset: 0b0x00000011111

Bit	Reset	Description
12	0x0	NON_SYM_EN: To enable non-symmetry mode 0 = DISABLE 1 = ENABLE
10:0	0x1f	CHANNEL_BIT_CNT: I2S, LJM, RJM mode: No. of bit clocks in left or right channel. DSP mode: No. of bit clocks in LEFT+RIGHT channel

27.2.6.4 I2S_I2S_FIFO_SCR_0

In the FIFO control/status register you can program required attention level for FIFO1 and FIFO2. Also figure out number of empty/full slots in FIFOs.

Clear the FIFOs by writing 1 in bits [8] and bit [12] for FIFO1 and FIFO2 respectively.

I2S Controller FIFO Control Register

Offset: 00ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxx0xxx0xx00xx00

Bit	R/W	Reset	Description
29:24	RO	X	FIFO2_FULL_EMPTY_COUNT: (Read Only) These bits indicate the number of full/empty slots in FIFO2 (full-count if bit-30=0; otherwise, empty-count)
21:16	RO	X	FIFO1_FULL_EMPTY_COUNT: (Read Only) These bits indicate the number of full/empty slots in FIFO1. (empty-count if bit-27=0; otherwise, full-count)
12	RW	0x0	FIFO2_CLR:Clear FIFO2. This bit gets cleared by hardware. 0 = NO_ACTION 1 = CLEAR

Bit	R/W	Reset	Description
8	RW	0x0	FIFO1_CLR:Clear FIFO1. This bit gets cleared by hardware. 0 = NO_ACTION 1 = CLEAR
5:4	RW	0x0	FIFO2_ATN_LVL: FIFO2 attention level. At this attention level, FIFO will generate request for read/write 0 = ONE_SLOT 1 = FOUR_SLOTS 2 = EIGHT_SLOTS 3 = TWELVE_SLOTS
1:0	RW	0x0	FIFO1_ATN_LVL: FIFO1 attention level. At this attention level, FIFO will generate request for read/write 0 = ONE_SLOT 1 = FOUR_SLOTS 2 = EIGHT_SLOTS 3 = TWELVE_SLOTS

27.2.6.5 I2S_I2S_PCM_CTRL_0

This register programs the controller to be operated in PCM mode. Following are the guidelines.

1. For PCM-mode to work, I2S_CTRL[11:10] should be programmed in DSP-mode.
2. Program bits-4 and 0 after programming all other bit-fields in this register.
3. Program the Transaction-enable bits (I2S_CTRL[29:28] after setting TRM/RVC modes in this register(bits 0 and 4).
4. Don’t over-write this register when the controller has any of the bits I2S_CTRL[31:28] asserted.
5. Program mask bits field to get required bit size(other than 16, 20, 24 and 32). For example, to get 13 bit size, BIT_SIZE field of I2S_CTRL[9:8] should be 0 (16 bit) and mask bits should be 3.

I2S Controller PCM Control Register

Offset: 010h | Read/Write: R/W | Reset: 0b00000000000

Bit	Reset	Description
10:9	0x0	PCM_TRM_EDGE_CNTRL: Highz control 0 = POS_EDGE_NO_HIGHZ 1 = POS_EDGE_HIGHZ 2 = NEG_EDGE_NO_HIGHZ 3 = NEG_EDGE_HIGHZ
8:6	0x0	TRM_MASK_BITS: Transmission mask bits 0 = ZERO 1 = ONE 2 = TWO 3 = THREE 4 = FOUR 5 = FIVE 6 = SIX 7 = SEVEN
5	0x0	FSYNC_PCM_CTRL: Short fsync (sync will be one-bit-clock wide). Long fsync (sync will be two-bit-clocks wide) 0 = SHORT 1 = LONG

Bit	Reset	Description
4	0x0	PCM_TRM_MODE: Bit format should be in DSP-format 0 = DISABLE 1 = ENABLE
3:1	0x0	RCV_MASK_BITS: Receive mask bits 0 = ZERO 1 = ONE 2 = TWO 3 = THREE 4 = FOUR 5 = FIVE 6 = SIX 7 = SEVEN
0	0x0	PCM_RCV_MODE: Bit format should be in DSP-format 0 = DISABLE 1 = ENABLE

27.2.6.6 I2S_I2S_NW_CTRL_0

This register programs the controller to be operated in Network mode. Following are the guidelines.

1. For Network-mode to work, I2S_CTRL[11:10] should be programmed in DSP-mode.
2. Program bits-3 and 0 after programming all other bit-fields in this register.
3. Program the Transaction-enable bits (I2S_CTRL[29:28] after setting TRM/RVC modes in this register(bits 0 and 3).
4. Don’t over-write this register when the controller has any of the bits I2S_CTRL[31:28] asserted.

I2S Controller Network Control Register

Offset: 014h | Read/Write: R/W | Reset: 0b000000

Bit	Reset	Description
5:4	0x0	TRM_TLPHY_SLOT_SEL: Transmission slot selection 0 = SLOT1 1 = SLOT2 2 = SLOT3 3 = SLOT4
3	0x0	TRM_TLPHY_MODE: Bit format should be in DSP-format 0 = DISABLE 1 = ENABLE
2:1	0x0	RCV_TLPHY_SLOT_SEL: Receive slot selection 0 = SLOT1 1 = SLOT2 2 = SLOT3 3 = SLOT4
0	0x0	RCV_TLPHY_MODE: Bit format should be in DSP-format 0 = DISABLE 1 = ENABLE

27.2.6.7 I2S_I2S_TDM_CTRL_0

Offset: 020h | Read/Write: R/W | Reset: 0b0xxxxx00x0x000x00000xx0000000000

Bit	Reset	Description
31	0x0	TDM_EN: TDM mode 0 = DISABLE 1 = ENABLE
25	0x0	TX_MSB_LSB: Configure to appear MSB or LSB first on sdata out 0 = MSB_FIRST 1 = LSB_FIRST
24	0x0	RX_MSB_LSB: Configure to appear MSB or LSB first on sdata out 0 = MSB_FIRST 1 = LSB_FIRST
22	0x0	TDM_EDGE_CTRL: Output highz control. When it is set, highz will be driven at output when slot is not active 0 = NO_HIGHZ 1 = HIGHZ
20:18	0x0	TOTAL_SLOTS: Number of slots per fsync 0 = ONE 1 = TWO 2 = THREE 3 = FOUR 4 = FIVE 5 = SIX 6 = SEVEN 7 = EIGHT
16:12	0x0	TDM_BIT_SIZE: Total bits per slot. It should be multiples of four - 1.
9:8	0x0	RX_DATA_OFFSET: Data offset to fsync 0 = ALIGNED_TO_FSYNC 1 = ONE_CLOCK 2 = TWO_CLOCKS 3 = THREE_CLOCKS
7:6	0x0	TX_DATA_OFFSET: Data offset to fsync 0 = ALIGNED_TO_FSYNC 1 = ONE_CLOCK 2 = TWO_CLOCKS 3 = THREE_CLOCKS
5:0	0x0	FSYNC_WIDTH: Fsync width interms of bit clocks.

27.2.6.8 I2S_I2S_TDM_TX_RX_CTRL_0

Offset: 024h | Read/Write: R/W | Reset: 0b0x0xxxxxxxxxxxx0000000000000000

Bit	R/W	Reset	Description
31	RW	0x0	TDM_TX_EN: TDM transmission enable 0 = DISABLE 1 = ENABLE
30	RO	X	TDM_TX_BSY: Transmission busy status bit 0 = NOT_BUSY 1 = BUSY
29	RW	0x0	TDM_RX_EN: TDM receive enable 0 = DISABLE 1 = ENABLE
28	RO	X	TDM_RX_BSY: Receive busy status bit 0 = NOT_BUSY 1 = BUSY

Bit	R/W	Reset	Description
15:8	RW	0x0	TDM_RX_SLOT_ENABLES: Rx Slot enables. Data will be received in enabled slots
7:0	RW	0x0	TDM_TX_SLOT_ENABLES: Tx Slot enables. Data will be transmitted in enabled slots

27.2.6.9 I2S_I2S_FIFO1_0

This register is used to write new PCM data samples to the controller (i.e. play samples). The status of the FIFO should be checked to ensure that a read will not cause an under run.

I2S Controller Output FIFO Buffer

Offset: 040h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	FIFO1_DATA: FIFO1 data

27.2.6.10 I2S_I2S_FIFO2_0

This register is used to read the contents of the PCM data IN (record) FIFO. The status of the FIFO should be checked to ensure that a write will not cause an overflow.

I2S Controller Input FIFO Buffer

Offset: 080h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	FIFO2_DATA: FIFO2 data

27.3 S/PDIF Controller

The S/PDIF controller supports both input and output in serial audio digital interface format. The input controller can digitally recover a clock from the received stream. The S/PDIF controller is also used to generate the embedded audio for HDMI output channel.

The controller conforms to the AES/EBU IEC 60958 standard.

The S/PDIF consists of two major modules:

- The **S/PDIF Output module**, responsible for sending data to the "spdifout" port in IEC 60958-3 bi-phase-mark code format
- The **S/PDIF Input module**, responsible for retrieving data to the "spdifin" port in IEC 60958-3 bi-phase-mark code format

Features

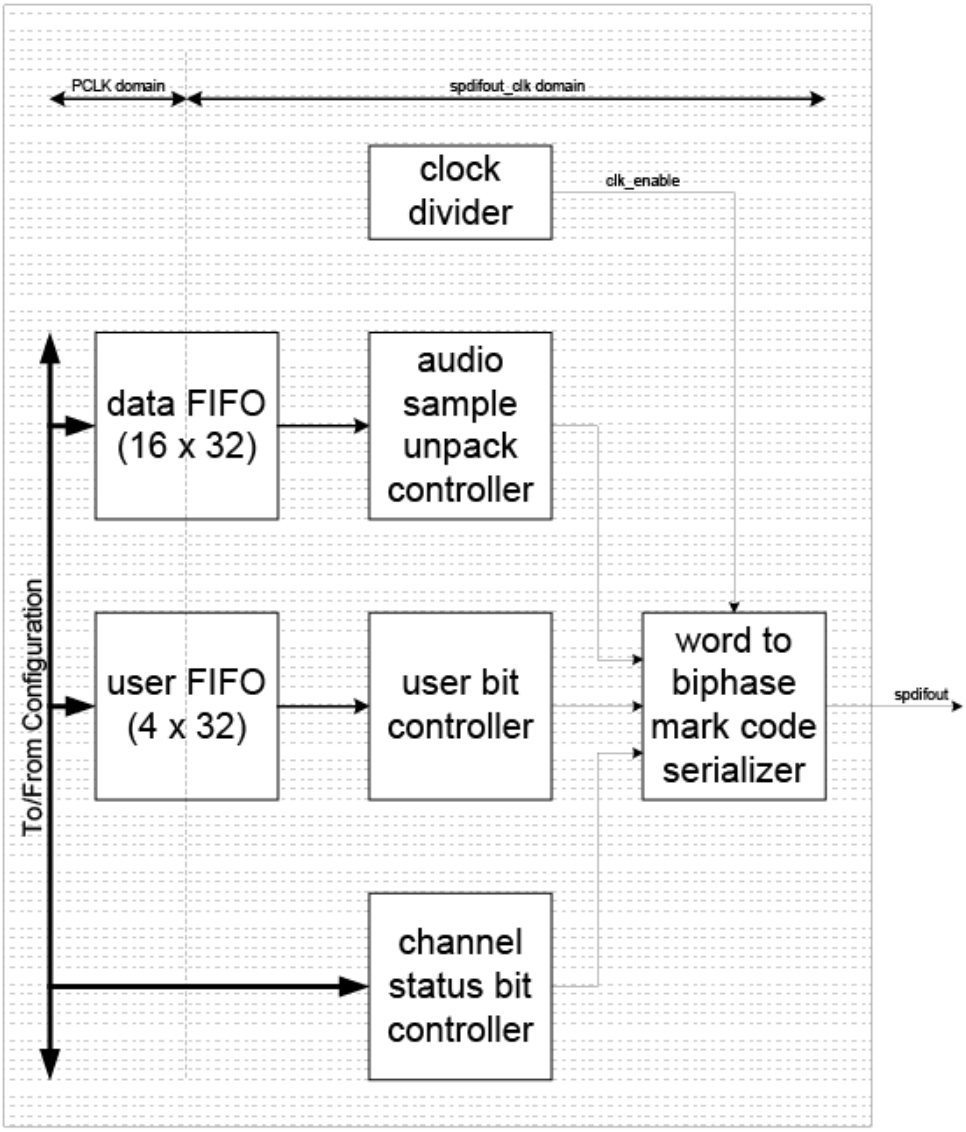
- Supports 5 data formats
  - 16-bit
  - 20-bit
  - 24-bit
  - Raw
  - 16-bit packed
- Supports "autolock" mode to automatically detect "spdifin" sample rate and lock onto the data stream.
- Supports "override" mode to provide a manual control to sample "spdifin" data stream.
- Provides a loopback mode to route the "spdifout" back to "spdifin" for self-testing.
- S/PDIF Output (transmitter)
  - 16-word data FIFO for storage of outgoing audio data
  - 4-word user FIFO for storage of outgoing user data
  - 6-word page buffer for storage of outgoing channel status
- S/PDIF Input (receiver)
  - 16-word data FIFO for storage of incoming audio data
  - 4-word user FIFO for storage of incoming user data
  - 6-word page buffer for storage of incoming channel status
- Maximum device clock of 50 MHz

27.3.1 S/PDIF Output Functionality

The S/PDIF Output module contains the following sub-blocks:

- A clock divider
- A 16-word data FIFO
- An audio sample unpack controller
- A 4-word user FIFO
- A user bit controller
- A channel status bit controller
- A word to bi-phase mark code serializer

Figure 80. S/PDIF Output Functional Block Diagram



Data FIFO

This 16-word deep FIFO is used to store audio samples written by the system in raw mode. It also stores non-audio data such as parity, channel status, user, validity, and preamble bits. This FIFO also serves as a bridge between the 'pclk' and 'spdifout_clk' domain.

Audio Sample Unpack Controller

Based on the PACK register field, this controller either breaks up the packed data from the data FIFO, or lets the Data FIFO's data flow through unmodified to the "word to bi-phase mark code serializer" module. This also controls when to pop an entry out of the data FIFO.

User FIFO

This 4-word deep FIFO is used to store user data written by the system. In raw mode, data from this FIFO is not used. This FIFO also serves as a bridge between the 'pclk' and 'spdifout_clk' domain.

User Bit Controller

When in 16, 20, 24-bit, or pack mode (PACK = 1 or BIT.MODE ≠11) and user data transmit is enabled (TXU.E = 1), this controller temporary stores an entry out of the user FIFO and feeds user data, at the rate of one bit per sub frame, to the "word to bi-phase mark code serializer" module. This controller also handles when to pop an entry out of the user FIFO. If user data transmit is disabled (TXU.E = 0), the user bit is forced to 0.

Channel Status Bit Controller

When in 16, 20, 24-bit, or pack mode (PACK = 1 or BIT.MODE ≠11) and channel status transmit is enabled (TXC.E = 1), this controller temporarily stores an entry out of the channel data TX page buffer and feed channel status at the rate of one bit per frame, to the "word to bi-phase mark code serializer" module. It also provides control to advance the channel data TX page buffer pointer to the next entry. If channel status transmit is disabled (TXC.E = 0), the channel status bit is forced to 0.

Word to Bi-Phase Mark Code Serializer

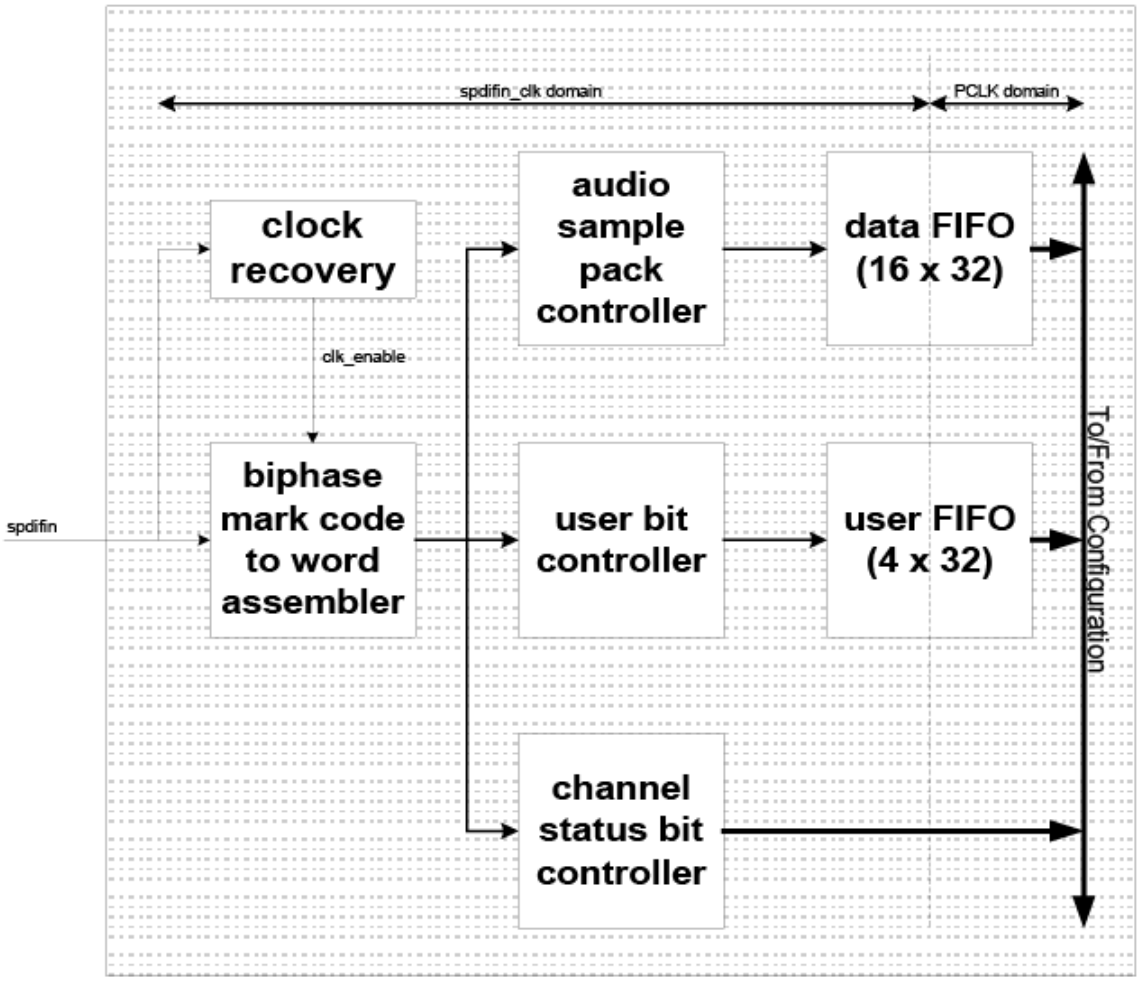
Based on the PACK and BIT.MODE register fields, this controller assembles a 32-bit word from the "Audio sample unpack controller" module, and/or "User bit controller" module, and/or "channel status bit controller" module. This 32-bit word is then converted to IEC 60958-3 bi-phase-mark code format and sent out through the 'spdifout' signal. This controller also provides various status signals (i.e. end of transmitting a sub frame, etc) back to the "unpack", "user", and "channel status" controllers.

27.3.2 S/PDIF Input Functionality

The S/PDIF Input module contains the following sub-blocks:

- A bi-phase mark code to word assembler
- An audio sample pack controller
- A 16-word data FIFO
- A user bit controller
- A 4-word user FIFO
- A channel status bit controller
- A clock recovery

Figure 81. S/PDIF Input Functional Block Diagram



Bi-Phase Mark Code to Word Assembler

Based on the clock enable pulse from the "clock recovery" module, this controller reads the 'spdifin' data stream in IEC 60958-3 bi-phase mark code format and converts/assembles bit data into a 32-bit word. This 32-bit word is then read by "audio sample pack controller", "user bit controller", and "channel status bit controller" modules for further processing. This controller also provides various status signals (i.e. end of receiving a subframe, etc) to the "pack", "user", and "channel status" controller.

Audio Sample Pack Controller

Based on the PACK register field, this controller either combines two audio data or lets the audio data flow through unmodified from the "bi-phase mark code to word assembler" module to the data FIFO. This also controls when to push an entry into the data FIFO.

Data FIFO

This 16-word deep FIFO is used to store audio samples written by the "audio sample pack controller" module. In 16, 20, 24-bit, or pack mode, it also stores non-audio data such as parity, channel status, user, validity, and preamble bits. This FIFO also serves as a bridge between the 'spdifin_clk' and 'pclk' domain.

User Bit Controller

This controller saves the user bit from each subframe and assembles them into a 32-bit word before pushing an entry into the user FIFO.

User FIFO

This 4-word deep FIFO is used to store user data written by the "user bit controller" module. This FIFO also serves as a bridge between the 'spdifin_clk' and 'pclk' domain.

Channel Status Bit Controller

This controller handles saving the channel status bit from channel A (subframe 1) and assembles them into a 32-bit word before writing into the 6-word channel status page buffer. The channel status page buffer also resides within this controller.

27.3.3 Programming Guidelines

27.3.3.1 16-, 20-, 24-bit Mode

During transmission, audio data is taken from the TXdata FIFO, channel status bit is taken from the TXchannel status page buffer, and the user status bit is taken from the TX user FIFO. The preamble bits are generated by the hardware. The valid bit is always zero.

During reception, audio data is stored in the RX data FIFO, 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 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.

27.3.3.2 Raw Mode

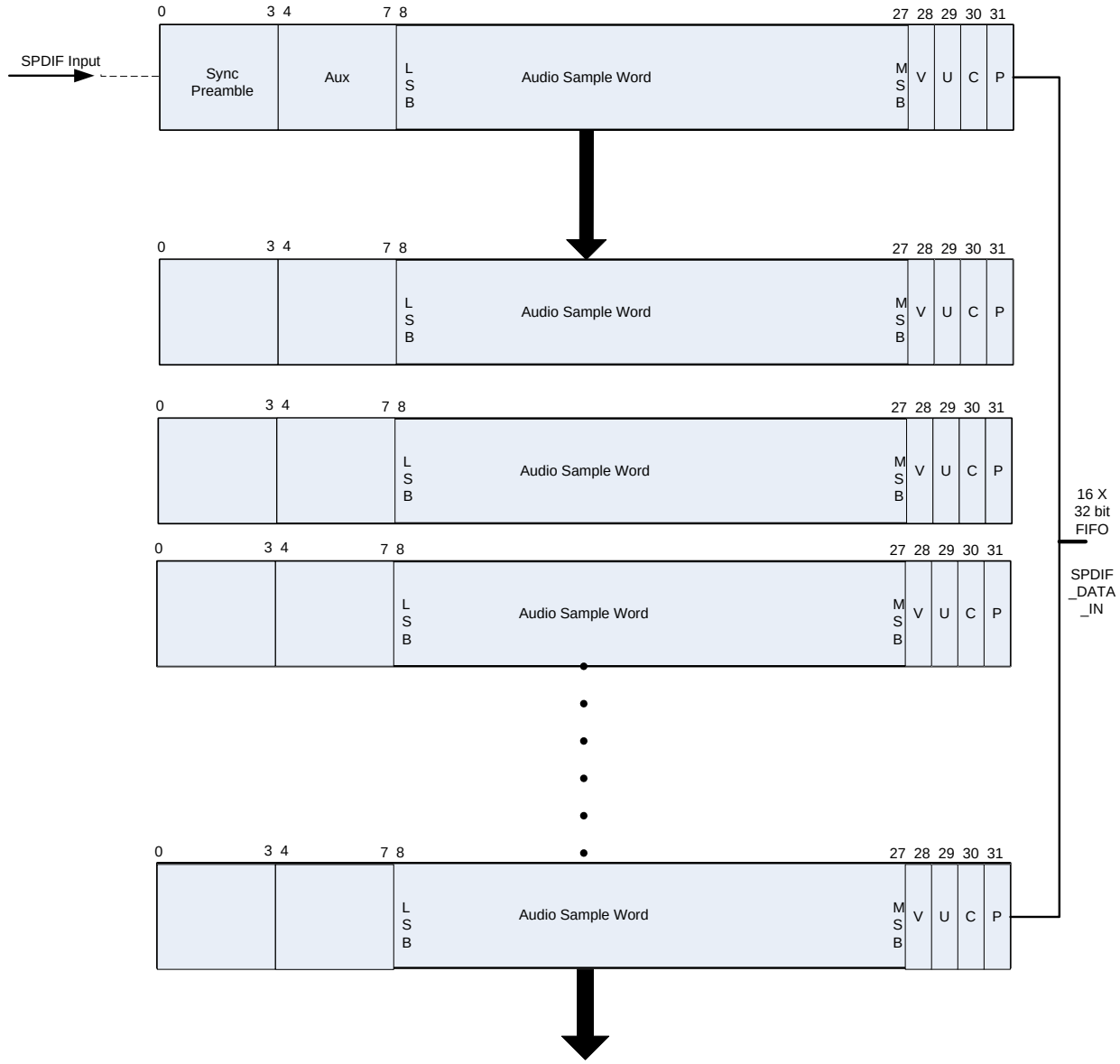
During transmission, the audio-data, preamble bits, channel status bits, user bits and valid bits are transmitted from the RX data FIFO. Firmware provides the correct preamble bits for the receiving device to synchronize with the transmitter. Firmware also provides preambles according to the following table:

Table 87. Raw Mode 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, preamble bits, channel status bits, user bits and valid bits are stored in the RX data FIFO. The user bit is also stored in the RX user FIFO and the channel bit is also stored in the RX channel status page buffer. But the channel status bits are stored only in the RX channel status page buffer after the B-preamble is detected.

Figure 82. S/PDIF Data Format for Raw Mode



27.3.3.3 Pack Mode

During transmission, audio data is taken from TX data FIFO which contains two 16-bit data (the upper 16-bit is for channel B or sub-frame 2, the lower 16-bit is for channel A or sub-frame 1), the channel status bits are taken from TX channel status page buffer and user status bits taken from TX user FIFO. The preamble bits are generated by the hardware. The valid bit is always zero.

During reception, audio data is stored in the RX data FIFO which contains two 16-bit data (the upper 16-bit being channel B data and the lower 16-bit being channel A data), the channel status bits are stored in the RX channel status page buffer and the user status bits stored in the RX user FIFO. The reception of the channel bits start after the B-preamble is detected.

27.3.3.4 Clock Source/Divider Control

The S/PDIF Output clock source/divider can be selected by programming CAR's CLK_SOURCE_SPDIF register. Refer to the registers below for the required input frequency for the desired output audio sample rate.

The S/PDIF Input clock source/divider can be selected by programming CAR's CLK_SOURCE_SPDIF register. Refer to the registers below for the required over-sample frequency.

27.3.3.5 Clock Enable Control

The application may enable the S/PDIF clocks by programming CAR's CLK_OUT_ENB_L_0 register.

27.3.3.6 Detecting Incoming Sampling Rate

The S/PDIF Input is capable of detecting and locking onto the 'spdifin' data stream regardless of the sample rate. It achieves this by using over-sampling technique.

In addition to reading the sampling rate from the channel status information, you can also read the PERIOD field in register STROBE_CTRL_0 to estimate the incoming sample rate. The last two digits of PERIOD indicate the fractional clock period.

Sample rate is approximately equal to ('spdifin' clock frequency * 1000) / (DECT_BI-PHASE_PERIOD * 128)

If using the PERIOD method, it is better to read this field towards the end of the audio file or just before turning off the S/PDIF Input. Although the hardware can lock onto the S/PDIF Input data stream in as little as 2 sub frames 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 bi-phase data stream. Over time, the strobe point varies less and less because the hardware has already adjusted to all the variations seen in the data stream.

27.3.3.7 S/PDIF Transmit Data Flow

Make sure the TX_FIFO is filled before enabling the transmission by setting TX_EN field of CTRL_0 register. Never allow the TX_FIFO to go empty at any time during transmission. The interrupt is generated when Tx DATA FIFO empty count is greater than the Tx DATA attention value (TX_ATN_LVL field of DATA_FIFO_CSR_0 register) if the Tx interrupt enable is set.

27.3.3.8 S/PDIF Receive Data Flow

To enable the reception of RX_FIFO set RX_EN field of CTRL_0 register. Do not allow the RX_FIFO to become full at any time during reception. The interrupt is generated when Rx DATA FIFO data count is greater than the Rx DATA attention value (RX_ATN_LVL field of DATA_FIFO_CSR register) if the Rx interrupt enable bit is set.

27.3.3.9 Filling/Retrieve Data from TX/RX User/Data FIFO

There are three methods of interfacing with the various FIFOs in S/PDIF. You may mix them to create the most efficient way of communicating with the FIFOs.

Method 1 (Preferred): DMA Method

Use DMA method by programming the APB-DMA registers. With this method, the FIFO attention levels need to be set just like in method 2. But instead of enabling interrupts when the FIFO attention level is reached, you should program the APB-DMA registers. Every time the attention level is reached, a DMA request will be generated by the S/PDIF to the APB-DMA and the APB-DMA will perform the data moving task from/to S/PDIF FIFOs to the AHB bus.

Method 2: Interrupt Method

Use interrupt method by setting the appropriate attention levels in the S/PDIF FIFO status register and enable the corresponding bits in the S/PDIF control register. When an attention level is reached, an interrupt will be signaled to ARM. Within the ISR, a query of the S/P-DIF interrupt sources can be found and the TX/RX user/data FIFOs can be filled or retrieved with appropriate amount of data.

Method 3: Polling Method

Use polling method by constantly checking the FIFOs' full/empty count from the S/PDIF DATA_FIFO_CSR register and fill/retrieve a certain amount of data to/from TX/RX user/data FIFOs.

27.3.4 Clock Rates

The table below shows the clock rates required by each sub-controller for the given sample rate.

Table 88. S/PDIF Clock Rates

Sample Rate (KHz)	S/PDIF Output clock freq (exact freq. MHz)	Period (ns)	S/PDIF Input clock freq (min. freq but <= 100 MHz)
32.0	4.0960	244.141	16.3840 (48 MHz typical)
44.1	5.6448	177.154	22.5790 (48 MHz typical)
48.0	6.1440	162.760	24.5760 (48 MHz typical)
88.2	11.2896	88.577	45.1584 (72 MHz typical)
96.0	12.2880	81.380	49.1520 (72 MHz typical)
176.4	22.5792	44.289	90.3168 (108 MHz typical)
192.0	24.5760	40.690	98.3040 (108 MHz typical)

27.3.5 S/PDIF Registers

27.3.5.1 SPDIF_CTRL_0

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 should not be attempted. It's important to fill transmitter's data/user FIFOs and channel status page buffer with valid data prior to enabling the transmitter (TX_EN=1) since the first FIFO/data buffer entries will be loaded immediately onto temporary storage buffers when TX_EN change to enable.

SPDIF Control Register

Offset: 000h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
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) 0 = DISABLE 1 = ENABLE
28	0x0	TX_EN: SPDIF Transmitter (TX) 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
27	0x0	TC_EN: Transmit Channel status. 0 = DISABLE 1 = ENABLE
26	0x0	TU_EN: Transmit user Data. 0 = DISABLE 1 = ENABLE
25	0x0	IE_TXE: Interrupt on transmit error. 0 = DISABLE 1 = ENABLE
24	0x0	IE_RXE: Interrupt on receive error. 0 = DISABLE 1 = ENABLE
23	0x0	IE_P: Interrupt on invalid preamble. 0 = DISABLE 1 = ENABLE
22	0x0	IE_B: Interrupt on "B" preamble. 0 = DISABLE 1 = ENABLE
21	0x0	IE_C: Interrupt when block of channel status received. 0 = DISABLE 1 = ENABLE
20	0x0	IE_U: Interrupt when a valid information unit (IU) receive. 0 = DISABLE 1 = ENABLE
19	0x0	QE_RU: Interrupt when RX user FIFO attn. level is reached. 0 = DISABLE 1 = ENABLE
18	0x0	QE_TU: Interrupt when TX user FIFO attn. level is reached. 0 = DISABLE 1 = ENABLE
17	0x0	QE_RX: Interrupt when RX data FIFO attn. level is reached. 0 = DISABLE 1 = ENABLE
16	0x0	QE_TX: Interrupt when TX data FIFO attn. level is reached. 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 bit needs to be padded to match the interface data bit size) 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
13:12	0x0	BIT_MODE: 00=16bit data 01=20bit data 10=24bit data 11=raw data 0 = MODE16BIT 1 = MODE20BIT 2 = MODE24BIT 3 = MODERAW

27.3.5.2 SPDIF_STATUS_0

IS_P, IS_B, IS_C, and IS_U are sticky bits. Software must write a 1 to the corresponding bit location to clear the status.

SPDIF Status Register

Offset: 004h | Read/Write: RO | Reset: 0bxxxxxxxxxxxx

Bit	Reset	Description
29	X	RX_BSY: Receiver (RX) shifter is busy receiving data. 1=busy, 0=not busy. This bit is asserted when the receiver first locked onto the preamble of the data stream after RX_EN is asserted. This bit is de-asserted when either, (a) the end of a frame is reached after RX_EN is de-asserted, or (b) the SPDIF data stream becomes inactive. 0 = FREE 1 = BUSY
28	X	TX_BSY: Transmitter (TX) shifter is busy transmitting data. 1=busy, 0=not busy. This bit is asserted when TX_EN is asserted. This bit is de-asserted when the end of a frame is reached after TX_EN is de-asserted. 0 = FREE 1 = BUSY
27	X	TC_BSY: TX is busy shifting out channel status. 1=busy, 0=not busy. This bit is asserted when both TX_EN and TC_EN are asserted and data from CH_STA_TX_A register is loaded into the internal shifter. This bit is de-asserted when either, (a) the end of a frame is reached after TX_EN is de-asserted, or (b) CH_STA_TX_F register is loaded into the internal shifter. 0 = FREE 1 = BUSY
26	X	TU_BSY: TX User data FIFO busy. 1=busy, 0=not busy. This bit is asserted when TX_EN and TXU_EN are asserted and there's data in the TX user FIFO. This bit is de-asserted when either, (a) the end of a frame is reached after TX_EN is de-asserted, or (b) there's no data left in the TX user FIFO. 0 = FREE 1 = BUSY
25	X	TX_ERR: Tx FIFO Under run error status 0 = NO_ERROR 1 = ERROR
24	X	RX_ERR: Rx FIFO Overrun error status 0 = NO_ERROR 1 = ERROR
23	X	IS_P: Preamble status: 1=bad/missing preamble, 0=Preamble ok 0 = OK 1 = MISSED
22	X	IS_B: B-preamble detection status: 0=not detected, 1=B-preamble detected 0 = NOT_DETECTED 1 = DETECTED

Bit	Reset	Description
21	X	IS_C: RX channel block data receive status: 1=received entire block of channel status, 0=entire block not received yet.
20	X	IS_U: RX User Data Valid flag: 1=valid IU detected, 0 = no IU detected.
19	X	QS_RU: RX User FIFO Status: 1=attention level reached, 0=attention level not reached.
18	X	QS_TU: TX User FIFO Status: 1=attention level reached, 0=attention level not reached.
17	X	QS_RX: RX Data FIFO Status: 1=attention level reached, 0=attention level not reached.
16	X	QS_TX: TX Data FIFO Status: 1=attention level reached, 0=attention level not reached.

27.3.5.3 SPDIF_STROBE_CTRL_0

SPDIF Data Strobe Control Register

Offset: 008h | Read/Write: R/W | Reset: 0bxxxxxxxx0xx00000xx000000

Bit	R/W	Reset	Description
23:16	RO	X	PERIOD: Indicates the approximate number of detected SPDIFIN clocks within a bi-phase period.
15	RW	0x0	STROBE: SPDIFIN Data Strobe Mode 1=Manual-locked strobe 0=Auto-locked strobe (default)
12:8	RW	0x0	DATA_STROBES: Manual data strobe time within the bi-phase clock period (in terms of the number of over-sampling clocks)
5:0	RW	0x0	CLOCK_PERIOD: Manual SPDIFIN bi-phase clock period (in terms of the number of over-sampling clocks)

27.3.5.4 SPDIF_DATA_FIFO_CSR_0

SPDIF FIFO Configuration and Status Register

Offset: 00ch | Read/Write: R/W | Reset: 0b00000000000001000000000000010000

Bit	Reset	Description
31	0x0	RU_CLR: Clear Receiver User FIFO (RX USR.FIFO)
30:29	0x0	RU_ATN_LVL: RX USR.FIFO Attention Level: 00=1-slot-full, 01=2-slots-full, 10=3-slots-full, 11=4-slots-full. 0 = RU1_WORD_FULL 1 = RU2_WORD_FULL 2 = RU3_WORD_FULL 3 = RU4_WORD_FULL
28:24	0x0	FULL_COUNT: Number of RX USR.FIFO levels with valid data.
23	0x0	TU_CLR: Clear Transmitter User FIFO (TX USR.FIFO)

Bit	Reset	Description
22:21	0x0	TU_ATN_LVL: TxUSR.FIFO Attention Level: 11=4-slots-empty, 10=3-slots-empty, 01=2-slots-empty, 00=1-slot-empty. 0 = TU1_WORD_EMPTY 1 = TU2_WORD_EMPTY 2 = TU3_WORD_EMPTY 3 = TU4_WORD_EMPTY
20:16	0x4	TU_EMPTY_COUNT: Number of Tx USR.FIFO levels that could be filled.
15	0x0	RX_CLR: Clear Receiver Data FIFO (RX DATA.FIFO).
14:13	0x0	RX_ATN_LVL: Rx FIFO Attention Level: 11=12-slots-full, 10=8-slots-full, 01=4-slots-full, 00=1-slot-full. 0 = RX1_WORD_FULL 1 = RX4_WORD_FULL 2 = RX8_WORD_FULL 3 = RX12_WORD_FULL
12:8	0x0	RX_DATA_FIFO_FULL_COUNT: Number of RX DATA.FIFO levels with valid data
7	0x0	TX_CLR: Clear Transmitter Data FIFO (TX DATA.FIFO)
6:5	0x0	TX_ATN_LVL: Tx FIFO Attention Level: 11=12-slots-empty, 10=8-slots-empty, 01=4-slots-empty, 00=1-slot-empty 0 = TX1_WORD_EMPTY 1 = TX4_WORD_EMPTY 2 = TX8_WORD_EMPTY 3 = TX12_WOR_DEMPTY
4:0	0x10	TD_EMPTY_COUNT: Number of Tx DATA.FIFO levels that could be filled.

27.3.5.5 SPDIF_DATA_OUT_0

Depends on BIT_MODE and PACK fields from CTRL register. The bits below have different definition.

- 16-bit mode (BIT_MODE=00, PACK=0).
  - [31:16] = 0.
  - [15:0] = transmit data [15:0].
- 20-bit mode (BIT_MODE=01, PACK=0).
  - [31:20] = 0.
  - [19:0] = transmit data [19:0].
- 24-bit mode (BIT_MODE=10, PACK=0).
  - [31:24] = 0.
  - [23:0] = transmit data [23:0].
- Packed 16-bit mode (BIT_MODE=00, PACK=1)
  - [31:16] = 16-bit right channel (channel B/sub frame 2).
  - [15:0] = 16-bit left channel (channel A/sub frame 1).
- Raw mode (BIT_MODE=11, PACK=0)
  - [31] = This bit carries the parity of bit[30:4] of the SPDIF data stream such that this bit together with bit[30:4] carry even parity.
  - [30] = This bit carries 1 bit of channel status information.
  - [29] = This bit carries 1 bit of user status information.

- [28] = This bit carries the validity bit associated with the main data field.
- [27:8] = These bits carry the audio sample word in linear 2's complement representation.
- [7:4] = These bits are the aux. bits.
- [3:0] = These bits are the preamble bits.

[3:2] = unused.

[1:0] = 00=undefined/invalid preamble, 01="B" preamble, 10="M" preamble, 11="W" preamble.

SPDIF Data Out Register

Offset: 040h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	P: Parity Bit (for raw mode).
30	0x0	C: channel status bit (for raw mode).
29	0x0	U: User data bit (for raw mode).
28	0x0	V: Valid bit (for raw mode).
31:16	0x0	TD_R15_TD_R0: Right channel audio data out (for packed 16-bit mode).
27:8	0x0	RAW_TD19_TXD0: 20 bit audio data (for raw mode).
7:4	0x0	TX_AX3_TX_AX0: Auxiliary data (for raw mode).
15:0	0x0	TD_L15_TD_L0: Left channel audio data out (for packed 16-bit mode).
15:0	0x0	TXD15_TXD0: channel audio data out (for 16-bit mode).
19:0	0x0	TXD19_TXD0: channel audio data out (for 20-bit mode).
23:0	0x0	TXD23_TXD0: channel audio data out (for 24-bit mode).
3:0	0x0	RAW_TX_PREAMBLE: Preamble data (for raw mode).

27.3.5.6 SPDIF_DATA_IN_0

Depends on BIT_MODE and PACK fields from CTRL register, the bits below have different definition.

- 16-bit mode (BIT_MODE=00, PACK=0).
  - [31] = This bit carries the parity of bit[30:4] of the SPDIF data stream such that this bit together with bit[30:4] carry even parity.
  - [30] = This bit carries 1 bit of channel status information.
  - [29] = This bit carries 1 bit of user status information.
  - [28] = This bit carries the validity bit associated with the main data field.
  - [27:24] = These are the preamble bits.

    [27] = 1=not parity error and "locked" onto the SPDIF data stream, 0=either parity error or not "locked" onto the SPDIF data stream.

    [26] = Indicate whether the preamble is positive or negative preamble. 0=positive, 1=negative.

    [25:24] = 00=undefined/invalid preamble, 01="B" preamble, 10="M" preamble, 11="W" preamble.
  - [23:16] = 0.
  - [15:0] = receive data [15:0].

- 20-bit mode (BIT_MODE=01, PACK=0).
  - [31] = This bit carries the parity of bit[30:4] of the SPDIF data stream such that this bit together with bit[30:4] carry even parity.
  - [30] = This bit carries 1 bit of channel status information.
  - [29] = This bit carries 1 bit of user status information.
  - [28] = This bit carries the validity bit associated with the main data field.
  - [27:24] = These are the preamble bits.

    [27] = 1=not parity error and "locked" onto the SPDIF data stream, 0=either parity error or not "locked" onto the SPDIF data stream.

    [26] = Indicate whether the preamble is positive or negative preamble. 0=positive, 1=negative.

    [25:24] = 00=undefined/invalid preamble, 01="B" preamble, 10="M" preamble, 11="W" preamble.
  - [23:20] = 0.
  - [19:0] = receive data [19:0].
- 24-bit mode (BIT_MODE=10, PACK=0).
  - [31] = This bit carries the parity of bit[30:4] of the SPDIF data stream such that this bit together with bit[30:4] carry even parity.
  - [30] = This bit carries 1 bit of channel status information.
  - [29] = This bit carries 1 bit of user status information.
  - [28] = This bit carries the validity bit associated with the main data field.
  - [27:24] = These bits are the preamble bits.

    [27] = 1=not parity error and "locked" onto the SPDIF data stream, 0=either parity error or not "locked" onto the SPDIF data stream.

    [26] = Indicate whether the preamble is positive or negative preamble. 0=positive, 1=negative.

    [25:24] = 00=undefined/invalid preamble, 01="B" preamble, 10="M" preamble, 11="W" preamble.
  - [23:0] = receive data [23:0].
- packed 16-bit mode (BIT_MODE=00, PACK=1)
  - [31:16] = 16-bit right channel (channel B/sub frame 2).
  - [15:0] = 16-bit left channel (channel A/sub frame 1).
- raw mode (BIT_MODE=11, PACK=0)
  - [31] = This bit carries the parity of bit[30:4] of the SPDIF data stream such that this bit together with bit[30:4] carry even parity.
  - [30] = This bit carries 1 bit of channel status information.
  - [29] = This bit carries 1 bit of user status information.
  - [28] = This bit carries the validity bit associated with the main data field.
  - [27:8] = These bits carry the audio sample word in linear 2's complement representation.
  - [7:4] = These bits are the aux. bits.
  - [3:0] = These bits are the preamble bits.

    [3] = 1=not parity error and "locked" onto the SPDIF data stream, 0=either parity error or not "locked" onto the SPDIF data stream.

    [2] = Indicate whether the preamble is positive or negative preamble. 0=positive, 1=negative.

    [1:0] = 00=undefined/invalid preamble, 01="B" preamble, 10="M" preamble, 11="W" preamble.

SPDIF Data Out Register

Offset: 080h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	P: Parity bit (for 16-bit/20-bit/24-bit/raw modes).
30	X	C: Channel Status bit (for 16-bit/20-bit/24-bit/raw modes).
29	X	U: User Data bit (for 16-bit/20-bit/24-bit/raw modes).
28	X	V: Valid bit (for 16-bit/20-bit/24-bit/raw modes).
27:24	X	RX_PREAMBLE: Preamble Code Symbol Bit (for 16-bit/20-bit/24-bit mode). [3]=Invalid Bit [2]=Polarity Bits [1:0]=Symbols B=01 W=11 M=10
31:16	X	RD_R15_RD_R0: Right channel audio data in (for packed 16-bit mode).
27:8	X	RAW_RXD19_RXD0: audio data in (for raw mode).
7:4	X	RX_AX3_RXAX0: auxiliary data in (for raw mode).
15:0	X	RD_L15_RD_L0: Left channel audio data in (for packed 16-bit mode).
15:0	X	RXD15_RXD0: audio data in (for 16-bit mode).
19:0	X	RXD19_RXD0: audio data in (for 20-bit mode).
23:0	X	RXD23_RXD0: audio data in (for 24-bit mode).
3:0	X	RAW_RX_PREAMBLE: Preamble Code Symbol Bit (for raw mode) [3]=Invalid Bit [2]=Polarity Bits [1:0]=Symbols B=01 W=11 M=10

27.3.5.7 SPDIF_CH_STA_RX_A_0

These 6-words receive channel data page buffer holds a block (192 frames) of channel status information. The order of receive is from LSB bit to MSB bit, and from CH_STA_RX_A to CH_STA_RX_F and back to CH_STA_RX_A.

Note: Only channel status bit from channel A (sub frame 1) will be saved into this page buffer.

SPDIF Channel Status Rx Page Buffer Register

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

Bit	Reset	Description
31:0	X	RX_C31_RX_C0: Channel status bits [31:0]; one bit per audio sample

27.3.5.8 SPDIF_CH_STA_RX_B_0

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

Bit	Reset	Description
31:0	X	RX_C63_RX_C32: Channel status bits [63:32]; one bit per audio sample

27.3.5.9 SPDIF_CH_STA_RX_C_0

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

Bit	Reset	Description
31:0	X	RX_C95_RX_C64: Channel status bits [95:64]; one bit per audio sample

27.3.5.10 SPDIF_CH_STA_RX_D_0

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

Bit	Reset	Description
31:0	X	RX_C127_RX_C96: Channel status bits [127:96]; one bit per audio sample

27.3.5.11 SPDIF_CH_STA_RX_E_0

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

Bit	Reset	Description
31:0	X	RX_C159_RX_C128: Channel status bits [159:128]; one bit per audio sample

27.3.5.12 SPDIF_CH_STA_RX_F_0

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

Bit	Reset	Description
31:0	X	RX_C191_RX_C160: Channel status bits [191:160]; one bit per audio sample

27.3.5.13 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” sub frames.

SPDIF Channel Status Tx Page Buffer Register

Offset: 140h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	TX_C31_TX_C0: Channel status bits [31:0]; one bit per audio sample

27.3.5.14 SPDIF_CH_STA_TX_B_0

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

Bit	Reset	Description
31:0	0x0	TX_C63_TX_C32: Channel status bits [63:32]; one bit per audio sample

27.3.5.15 SPDIF_CH_STA_TX_C_0

Offset: 148h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	TX_C95_TX_C64: Channel status bits [95:64]; one bit per audio sample

27.3.5.16 SPDIF_CH_STA_TX_D_0

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

Bit	Reset	Description
31:0	0x0	TX_C127_TX_C96: Channel status bits [127:96]; one bit per audio sample

27.3.5.17 SPDIF_CH_STA_TX_E_0

Offset: 150h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	TX_C159_TX_C128: Channel status bits [159:128]; one bit per audio sample

27.3.5.18 SPDIF_CH_STA_TX_F_0

Offset: 154h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	TX_C191_TX_C160: Channel status bits [191:160]; one bit per audio sample

27.3.5.19 SPDIF_USR_STA_RX_A_0

This 4-words deep FIFO receives user FIFO field information. The order of receive is from LSB bit to MSB bit.

Note: The user data from this FIFO will be used only when PACK=1, or when BIT_MODE=00/01/10. Each user bit will receive from each sub frame.

SPDIF User Data Input FIFO Register

Offset: 180h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	RX_U31_RX_U0: 32 bit user data input

27.3.5.20 SPDIF_USR_DAT_TX_A_0

This 4-words deep FIFO transmits user FIFO field information. The order of transmission is from LSB bit to MSB bit.

> **Note:** The user data from this FIFO will be used only when PACK=1, or when BIT_MODE=00/01/10. Each user bit will be sent out to "each" sub frame.

SPDIF User Data Output FIFO Register

Offset: 1c0h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	TX_U31_TX_U0: 32 bit user data output

28.0 CAMERA SERIAL INTERFACE (MIPI-CSI)

The Camera Serial Interface (CSI) is based on MIPI CSI 2.0 standard specification and implements the CSI receiver which receives data from an external camera module with a CSI transmitter. It consists of two CSI receiver interfaces so it can receive serial transmission from two cameras:

- CSI_A interface consisting of 1 clock lane and 2 data lanes.
- CSI_B interface consisting of 1 clock lane and 1 data lane

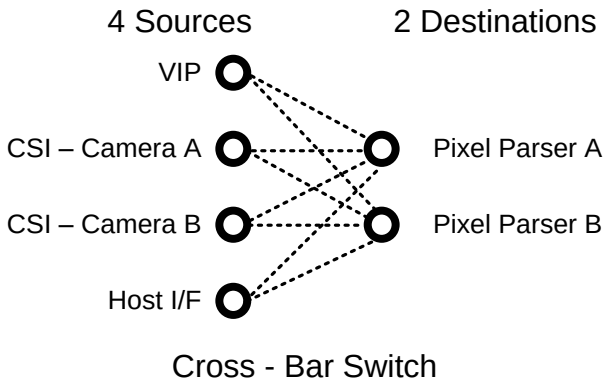
The CSI interface can also receive a digital CSI compatible byte stream from either the 8-bit Video Input (VI) port or from the host interface. The CSI stream coming from VI port is provided to accommodate the possibility of using an external CSI receiver or to take advantage of the rich CSI data formats transmission using the traditional 8-bit digital CMOS video interface. Similarly, CSI stream coming from host is provided to take advantage of the rich CSI data formats.

Only 1 data/pixel stream can be processed at any given time. The two streams can come from any one or two of the four possible sources, as shown in Figure 83. If the two streams come from a single source, then the streams are separated using a filter indexed on different virtual channel numbers or data types. In case of separation using data types, the normal data type is separated from the embedded data type. Since there are only two pixel parsers, virtual channel and embedded data capability cannot be used at the same time.

CSI supports both single-shot mode and continuous mode.

CSI is designed with error resilience.

Figure 83: Data Source/Destination Options



The MIPI CSI 2.0 standard is available from MIPI to its members at http://www.mipi.org/

28.1 Use Cases

The use cases for CSI fall into the following categories listed in Table 89:

Table 89: CSI Use Cases

Case	Description	Sources	Destinations
Normal	Standard CSI	Camera A or B	PPA or PPB
Virtual Channel	One stream goes to 2 parsers. Packet filtering based on VC number in header	Camera A,B VIP, or Host	PPA and PPB

Case	Description	Sources	Destinations
Embedded Data	One stream goes to 2 parsers. Packet filtering based on data type in header	Camera A,B VIP, or Host	PPA and PPB
Embedded Data (w/o filtering)	One stream goes to 1 parser. Has normal and embedded data	Camera A,B VIP, or Host	PPA or PPB
VIP	CSI payload-only or packet sources from VIP (CYA mode) *	VIP	PPA or PPB
Host	Payload-only or Packet sources from Host I/F	Host I/F	PPA or PPB

* In the Tegra® 2 Processor, packet mode in the CSI VIP path is not supported.

28.2 Input Data Format

The supported data formats are summarized in Table 90.

Table 90: Supported CSI Formats

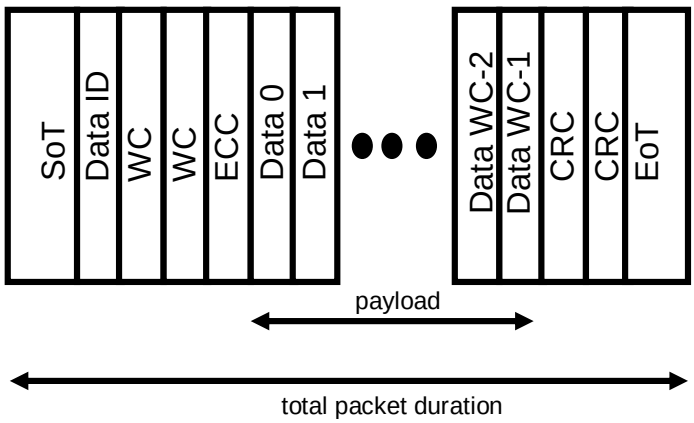
Data Format	
RGB	RGB888, RGB666, RGB565, RGB555, RGB444
YUV	YUV422-8b, YUV422-10b, YUV420-8b (legacy), YUV420-8b, YUV420-10b, YUV444-8b
RAW	RAW6, RAW7, RAW8, RAW10, RAW12, RAW14
User defined	JPEG8
Embedded	Embedded control information

The formats YUV422/420-10b are converted to YUV422/420-8b by taking the most significant bits.

28.3 CSI Packet Structure

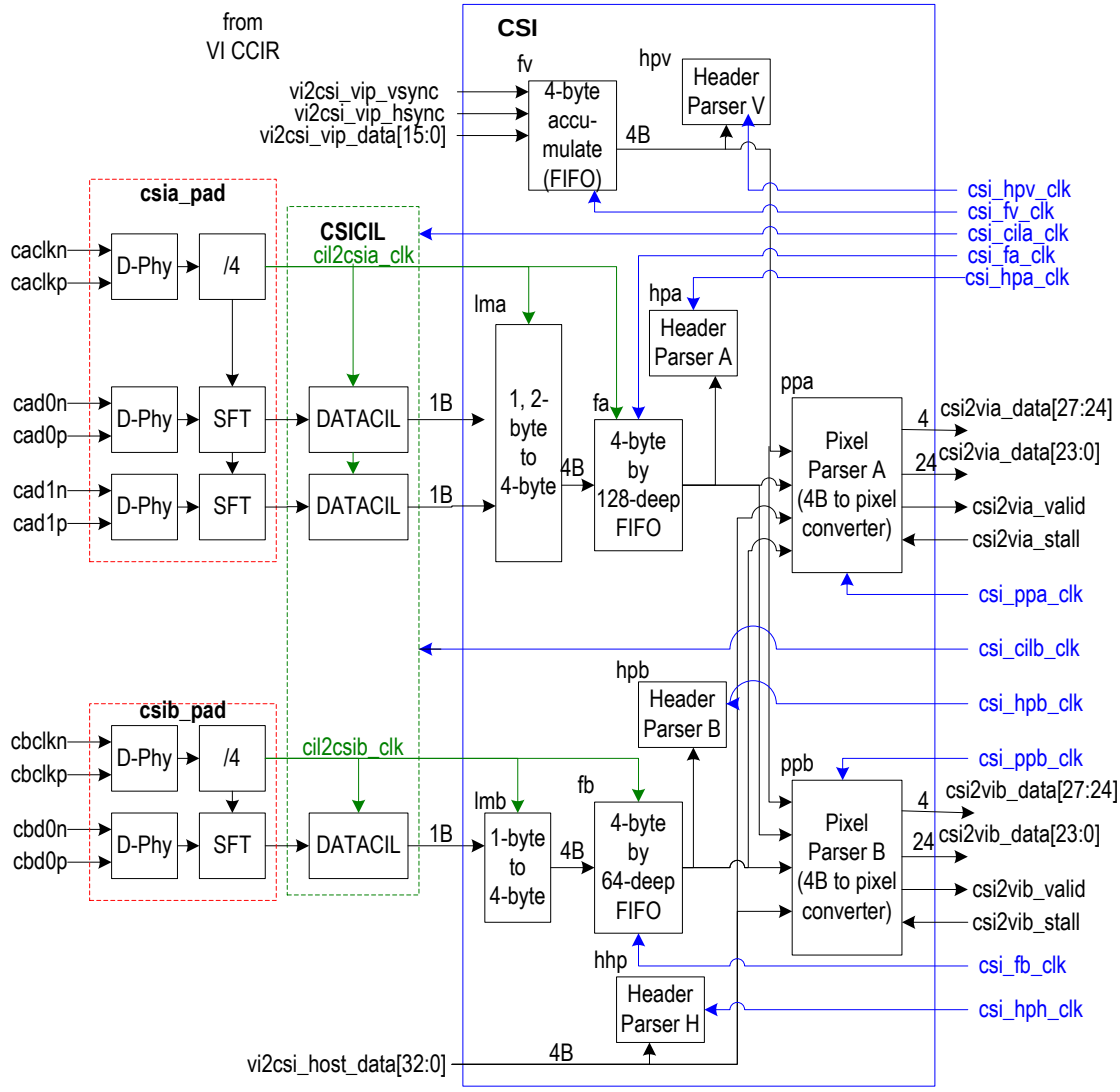
The CSI packet structure is illustrated in Figure 84. It has a start-of-transmission sequence, SoT, which contains a 1-byte preamble sequence. The packet is composed of a 4-byte header, a body of WC bytes, and a 2-byte CRC check. The end-of-transmission, EoT, is indicated by the line state going from high speed transmission to LP11 state.

Figure 84: CSI Packet Structure



28.4 CSI Implementation

Figure 85: CSI Block Diagram



Clocks highlighted in blue are 2nd level gated clock that use viclk as root clock.
Clocks highlighted in green are byte clocks that are generated by the CIL pad cells.

The CSI interface essentially consists of:

- MIPI D-Phy block
- Control Interface Logic (CIL) for Serial Clock Receiver
- CSI data path module which is implemented as a sub-module of VI module

In Tegra® 2, two CSI bricks including csia_pad with 2x data lanes and csib_pad with 1x data lane receive serial data from cameras, de-serialize them into 8b parallel data, and send them to the CSICIL block. The CSICIL block performs packet boundary detection by searching for the packet preamble. The main CSI module receives 8b packet aligned data from CSICIL, performs parity checking on the header, header decoding, CRC check, and pixel parsing. The output is sent out to VI through the 28b bus which outputs 1 pixel per VI clock.

28.5 Performance Limitations

Performance of the CSI interface is subject to several factors that must all be met. The performance factors that are directly related to the CSI interface are the serial data rate and the internal pixel rate. Maximum serial data rate is expected to be 1Gbps given the ideal system condition. This data rate may be degraded depending on system design and may also be limited by the CSI transmitter in the camera.

For each CSI data lane, the serial data stream received by the CSI PHY is internally converted to a byte stream by the CSI Control Interface Logic (CIL) before further processed by the CSI module that resides as part of the Video Input (VI) module. The CSI byte clock is used to transfer this data byte stream (1 byte per data lane) between the CIL and the CSI data paths. This CSI byte clock is derived from the received CSI serial clock by dividing the CSI serial clock by 4. So for 1Gbps serial CSI data rate, the serial clock frequency is 500 MHz and the byte clock frequency is 166 MHz.

The byte stream received by the CSI CIL logic is converted to a pixel stream at 1 pixel per clock in the CSI module. This pixel stream output is further processed by other modules (ISP or the rest of VI datapath or EPP). The maximum output pixel clock rate of CSI module is 166 MHz.

Depending on where the CSI pixel stream is processed (ISP, VI, or EPP), the maximum pixel clock frequency is limited by the pixel clock frequency of the modules that process this pixel stream.

Note:	VI/ISP/EPP pixel frequency may be lower than CSI output pixel clock frequency and therefore, may limit the actual pixel data rate. Please refer to ISP, VI and EPP specification.

Note that with multiple data lanes per CSI interface, the maximum serial data link speed may not be achievable due to pixel clock frequency limitations. Also, if two data streams come from the same CSI interface, since the streams are interleaved in time, the pixel clock frequency limitations may also limit the frame rate of the combined streams.

Table 91: Packet Efficiency

	Best* WC=65536	Typical* WC=1024
Payload	524us	8192ns
Overhead	356ns	356ns
Efficiency	99.9%	96%

28.6 Error Resilience

CSI is required to be designed to tolerate and/or recover stream errors at various levels.

According to CSI 2.0 spec, CSI must be able to detect and correct one-bit header error, detect two or more bits header errors, and detect possible payload errors using 16-bit CRC.

Due to the architectural constraints in VI and EPP, CSI has to always send “valid” stream data to VI, even when the incoming stream contains errors and causes CSI to drop packets. For stream data with a format other than *embedded*, “valid” stream data means one or more full frames of data that VI exactly expects for certain frame width and frame height.

If the incoming stream data (non-embedded) contains less bytes than the word count (WC), or less number of lines than the expected frame height, CSI must be able to pad pixels to short lines, or pad lines to short frames, so that CSI always sends “valid” frame data to VI.

28.7 Other Architectural Constraints

The following specifies other architectural constraints for this design:

- Because there are only 2 pixel parsers, embedded data and virtual channel cannot be supported simultaneously.
- Embedded data in a frame can be output in the same output port as the pixel data stream; however, in some cases where image processing or data reformatting is performed in VI/ISP/EPP, this embedded data may be accidentally processed in VI/ISP/EPP. Currently, VI/ISP/EPP cannot differentiate between embedded data or pixel data.
- CSI stream from VI port is limited to VI port input clock frequency (96 MHz) at 8-bit/clock. This stream may have fully compatible CSI packets format or it may carry only CSI compatible data format without packet header/footer and short packets. If it carries only CSI compatible data format, it should be transmitted using VI video input stream format with external H/V syncs.
- CSI stream from host interface should be sent in 32-bit data through the Y-FIFO with frame/line definition compatible only. So, CSI stream coming from host should only contain CSI compatible data formats but without the CSI packet header/footer or short packets. The header information should be set using registers.
- Same stream going to different pixel parsers. This case will work as long as no stall comes from VI side. If stall comes then the current frame will be invalid. This is an exception case which will never normally occur. STM_ERR will be generated and S/W needs to take appropriate action.

28.8 CSI Datapath Module

The CSI datapath consists of four input ports and two output ports. There are two datapaths for pixel stream processing; therefore, at any time, a maximum of two input ports maybe active simultaneously.

The components of CSI datapath include:

- Four input ports from: CSI A interface, CSI B interface, 8-bit VI port, and 32-bit host interface. Each port is capable of carrying a stream with CSI compatible data format.
- Three asynchronous input buffers (FIFOs) to receive streams from VI port, CSI A interface, and CSI B interface.
- Four header parsers for searching packet header and to perform error detection and correction of CSI header.
- Two pixel parsers with corresponding output ports. Each pixel parser can convert CSI packet data stream to an output a pixel stream. Each output port consists of maximum 24-bit of data per clock. Depending on the output data format options, one or two pixels per clock may be sent.

28.8.1 Header parser

CSI pixel stream processing mainly consists of header parser and pixel parser. There are 4 header parsers: header parser A, header parser B, header parser V, and header parser H.

Header parser A is enabled when CSI A interface is selected as input source. Similarly header parser B is enabled when CSI B interface is selected as input source.

In general the header parser is used for:

- Detecting long and short packet headers and deciding what to do with the packet. This includes performing error detection and correction on the packet header prior to making decision and dealing with uncorrectable packet header.
- Skipping extra data on longer than expected data packets.
- Skipping next packet on imminent input buffer overflow when the pixel parser is busy processing current packet.

When enabled, the header parser will interpret CSI packet header, perform error detection and correction. If the packet header is uncorrectable, then an error is flagged and interrupt generated. If good or correctable CSI packet header is found, then the header parser will check the Data Identifier (DI) byte in the packet header and check the 2-bit Virtual Channel (VC) identifier and the 6-bit Data Type (DT) within this Data Identifier byte and will also check the 2-byte Word Count (WC) value in the

packet header. If the virtual channel ID does not match the expected (programmed) value for the corresponding pixel parser then the packet is discarded. If the virtual channel ID matches the expected (programmed) value for the corresponding pixel parser then the header parser must decide what to do with the packet depending on the Data Type and the Word Count value. For long packet, if the Data Type does not match the expected (programmed) value for the corresponding pixel parser then the packet is discarded. If the Data Type of the long packet matches the expected (programmed) value for the corresponding pixel parser, then the corresponding pixel parser is notified to process the remaining packet data and checksum.

When first enabled, header parser must always search for Frame Start packet which is a CSI short packet. However, when input source is VI or host, there is an option to indicate Frame Start using the incoming vertical sync signal. If vertical sync signal is used to indicate frame start then if CSI frame start packet is encountered then the header parser will discard it. When frame start packet/signal is found, the header parser will notify the corresponding pixel parser so that it can make preparation to process a new frame and output frame start code.

28.8.1.1 Data Type

There are 64 possible types based on 6 bit Data Type value.

Data Type	Description
0x00 – 0x07	Synchronization Short Packet Data Types
0x08 – 0x0F	Generic Short Packet Data Types
0x10 – 0x17	Generic Long Packet Data Types
0x18 – 0x1F	YUV Data
0x20 – 0x27	RGB Data
0x28 – 0x2F	RAW Data
0x30 – 0x37	User Defined Byte-based Data
0x38 – 0x3F	Reserved

28.8.1.2 Short packets

There are 16 possible short packets based on Data Type value.

Data Type	Description
0x00	Frame Start Code
0x01	Frame End Code
0x02	Line Start Code (Optional)
0x03	Line End Code (Optional)
0x04 – 0x07	Reserved

28.8.1.3 Long packets

CSI long packets are data packets which always consist of 1-line of data per packet with exception of arbitrary data packets. There are various data types defined. Please refer to Section 28.2 in this document for a summary of CSI data formats and their corresponding CSI module internal output format.

Generic long packets need special handling. There are 3 generic long packet types defined: null packet (DT=0x10), blanking data packet (DT=0x11), and embedded data packet (DT=0x12). There are also 5 reserved generic long packet types (DT=0x13 to 0x17). The header parser should discard all null packets and all reserved generic long packets.

Blanking data packets may contain blank color or blank image information. The transmission of blanking data packet is optional in the CSI2.0 specification. Null and blanking data are defined in CSI2.0 specification mainly to accommodate a receiver which is timing sensitive and requires line start/end for every line in the frame. As a CSI receiver, this product does not have such blank timing sensitivity and it typically does not process blank data and does not store it to memory. Blanking data packet can therefore be discarded by the header parser.

However, some module may have a requirement for specific number of “blank” or extra lines at the end of vertical active area, such as the ISP module, which requires input active scan area to be about 6 lines larger than output active area. If a module that takes CSI output stream requires extra lines at the end of active image then blanking packet data can be used to generate these additional lines. In the future, there might be other reasons to process blank data. So, an option should be provided to accept and process blanking data. In the absence of better definition of blanking data format in the CSI spec, if software decides that blanking data cannot be discarded then, the pixel parser should assume that the blanking data format is the same as the programmed expected data type of the pixel stream.

The current 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 as 8-bit/clock or 16-bit/clock format. If there are two video sources that occupy both the pixel parsers, then embedded data if it has to be stored to memory, needs to be output in the same channel as the image data as 8-bit/16-bit data. But this also means that the module that receives output of CSI module and does image processing must have the ability to differentiate embedded data and not do image processing on this data prior to writing it to memory.

28.8.1.4 Top or Bottom Field Tag

During frame start, a tag is passed from CSI to VI to indicate top or bottom field. The 1-bit tag is at the least significant bit of the CSI output bus. If the bit is “1”, the field is top, but bottom otherwise.

28.9 Software Requirements

28.9.1 Error Counters

To help debug, three 32-bit error counters have been implemented. SW can program each counter to increment at the event of any one of many error conditions or status change. The error conditions are:

- Header errors corrected
- Header uncorrectable errors
- CRC errors
- Packets too short, actual length less than WC
- Overflow errors due to padding packets too short
- FIFO overflow errors
- Illegal word count
- SoT single-bit error
- SoT multi-bit error, packet discarded
- EoT sequence error
- ESC-entry error
- LP-CTRL error
- The statistics that can be monitored include:
- line packets processed

- total packets processed
- ESC command processed

28.9.2 Programming Sequence

The following sequence is recommended for capturing a single frame:

1. Setup 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 thru I2C interface
5. All CSI data and clock lanes are in stop state, LP11
6. Initiate frame capture thru I2C
7. Frame done, CSI goes back to stop state, LP11

28.9.3 Escape Mode Handling

In the MIPI PHY specification, an escape mode is available for low speed command and data transfer. In our implementation, the escape mode entry is detected inside CSICIL. The command byte will be deposited in a register and interrupt generated from CSI. Software reads the register and decodes. CSI supports only one escape mode command which is Ultra-Low Power Mode (ULPM).

ULPM command

Upon receiving the ULPM, software will synchronously power down the CSI interface and the camera as follows:

1. Software put camera in sleep mode thru I2C
2. Camera sends ULPM command to CSI
3. CSI receives ULPM and raise interrupt
4. Software powers down CSI which goes to sleep in sync with camera’s CSI transmitter

28.10 DPHY Modes of Operation

The MIPI DPHY has 3 basic modes of operation. High speed mode employs differential signaling and achieves data rate of 1Gbps. Both the driver and receiver are matched to 100 ohm differential. The driver is voltage mode for lower power consumption as opposed to current mode.

Low power control mode employs single-ended CMOS signaling for handshaking between camera and host. In this mode, there is no clock and no maximum symbol time defined in the specification. In the NV implementation, the receiver samples the input using both edges of the internal pixel clock, so the timing resolution is half the clock period of the pixel clock period.

The low power escape mode employs self-clocking using both data P and N pins. The clock is generated by XOR of data P and N. The maximum transfer rate is 20 Mbps for a 50ns bit time. In our implementation, the receiver samples the input using both edges of the internal pixel clock also. It is the goal to sample at least twice during the bit interval; therefore, the internal pixel clock should have a minimum frequency of 20 MHz.

Table 92: MIPI DHY Mode of Operations

Modes	Description	Clock
High speed (HS)	High speed differential signaling. Upto 1 Gbps. Burst transmission for low power.	500 MHz differential
Low Power (LP) Control	Single-ended 1.2V CMOS level. Low speed signaling for handshaking. Supports ULPM sleep state.	No Clock
Low Power (LP) Escape	Same as above. Low speed signaling for data, used for escape command entry only. 20Mbps	2 stage synchronizers to VI clock

28.11 MIPI-CSI Registers

28.11.1 CSI_VI_INPUT_STREAM_CONTROL_0

VI Input Stream Control

Offset: 200h | Read/Write: R/W | Reset: 0bx

Bit	Reset	Description
7	X	VIP_SF_GEN: VIP Start Frame Generation Don't use vi2csi_vip_vsync to generate start frame (SF), or end frame (EF) markers in the pixel parser output stream. 0 = VSYNC_SF : Pulses on vi2csi_vip_vsync will be used to generate start frame (SF) and end frame (EF) markers in the pixel parser output stream. In Tegra 2, only payload_only mode is supported in the VIP input stream path, and this field may always be programmed to VSYNC_SF. 1 = NO_VSYNC_SF

28.11.2 CSI_HOST_INPUT_STREAM_CONTROL_0

Host Input Stream Control

Offset: 202h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
28:16	RW	X	HOST_FRAME_HEIGHT: Host Frame Height Specifies the height of the host frame when the host is supplying CSI format payload only data to one of the CSI pixel parsers. Programmed Value = number of lines - 1
8	RO	X	HOST_END_OF_PACKET: Writing this bit with a 1 indicates End of Packet, when CSI Host data is being received in Packet Format. In Packet Format vi2csi_host_hsync is not used to indicate beginning of packet.
7	RW	X	HOST_SF_GEN: Host Start Frame Generation Don't use CSI Host Line counter to generate start, or End, of Frame control outputs. This setting should only be used if HOST_DATA_FORMAT is set to PACKETS, and the Host data stream has frame sync packets. 0 = LINE_COUNTER : CSI Host Line counter will be used to generate Frame start and end control. To signal the start of the first frame the pixel parser will send a SF control, and signal start of frame mark, when it is first enabled with Host as its source. This setting should be used when HOST_DATA_FORMAT is set to PAYLOAD_ONLY. 1 = SHORT_PACKETS
3:0	RW	X	HOST_DATA_FORMAT: Host Data Format Data written to Y_FIFO_WRITE port should be in CSI packet format. To indicate end of packet a 1 should be written to HOST_END_OF_PACKET. A 1 should also be written to HOST_END_OF_PACKET before writing the first word of packet data to Y_FIFO_WRITE. 0 = PAYLOAD_ONLY : Data written to Y_FIFO_WRITE port should be CSI line payload data only (no header, no footer, and no short packets). A value of 1 should not be written to HOST_END_OF_PACKET (end of packet pulse only gets generated when a 1 is written to this bit). First line will be indicated when one of the pixel parsers is first enabled with its CSI_PPA/B_STREAM_SOURCE set to "HOST". The values in the following

Bit	R/W	Reset	Description
			PIXEL_STREAM_A/B_CONTROL0 fields, for the pixel parser that is receiving host data, will be ignored; CSI_PPA/B_PACKET_HEADER overridden with "NOT_SENT", CSI_PPA/B_DATA_IDENTIFIER overridden with "DISABLED", CSI_PPA/B_WORD_COUNT_SELECT overridden with "REGISTER". CSI_PPA/B_CRC_CHECK overridden with "DISABLE", CSI_PPA/B_VIRTUAL_CHANNEL_ID, CSI_PPA/B_EMBEDDED_DATA_OPTIONS, and CSI_PPA/B_HEADER_EC_ENABLE. CSI_PPA/B_DATA_TYPE should be programmed with the 6 bit data type that is to be used to interpret the stream. CSI_PPA/B_WORD_COUNT should be programmed with the number of bytes per line. 1 = PACKETS

28.11.3 CSI_INPUT_STREAM_A_CONTROL_0

CSI Input Stream A Control

Offset: 204h | Read/Write: R/W | Reset: 0b01111111xxxxxxxxxxx0xx01

Bit	Reset	Description
23:16	0x7f	CSI_A_SKIP_PACKET_THRESHOLD: CSI-A Skip Packet Threshold. This value is compared against the internal FIFO that buffer the input streams. A packet will be skipped (discarded) if the pixel stream processor is busy (probably due to padding process of a short line) and the number of entries in the internal FIFO exceeds this threshold value. Note that each entry in the internal FIFO buffer is four bytes. To turn off this feature, set the value to its maximum value (all ones).
4	0x0	CSI_A_SKIP_PACKET_THRESHOLD_ENABLE: Enables skip packet threshold feature. Skip packet feature is enabled. 0 = DISABLE : Skip packet feature is disabled. 1 = ENABLE
1:0	0x1	CSI_A_DATA_LANE: CSI-A Data Lane 0= 1 data lane 1= 2 data lanes 2= 3 data lanes 3= 4 data lanes

28.11.4 CSI_PIXEL_STREAM_A_CONTROL0_0

CSI Pixel Stream A Control 0

Offset: 206h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000 |

Bit	Reset	Description
29:28	X	CSI_PPA_PAD_FRAME: CSI Pixel Parser A Pad Frame This specifies how to deal with frames that are shorter (fewer lines) than expected. Short frames are usually caused by line packets being dropped because of packet errors. Expected frame height is specified in PPA_EXP_FRAME_HEIGHT. To do padding the value in CSI_PPA_WORD_COUNT needs to be set to the number of input bytes in each line's payload. Short frames will not be padded out. 0 = PAD0S : Lines of all zeros will be used to pad out frames that are shorter than expected height. PPA_EXP_FRAME_HEIGHT must be programmed to an appropriate value if this fields is set to PAD0S. 1 = PAD1S : Lines of all ones will be used to pad out frames that are shorter than expected height. PPA_EXP_FRAME_HEIGHT must be programmed to an appropriate value if this fields is set to PAD1S. 2 = NOPAD
27	X	CSI_PPA_HEADER_EC_ENABLE: CSI Pixel Parser A Packet Header Error Correction Enable. This parameter specifies whether single bit errors in the packet header will be automatically corrected, or not. Single bit errors in the header will not be corrected. Header ECC check will still set header ECC status bits and the packet will be processed by Pixel Parser A. DISABLE should not be used when processing interleaved streams (Same stream going to both PPA and PPB). 0 = ENABLE : Single bit errors in the header will be automatically corrected. 1 = DISABLE
25:24	X	CSI_PPA_PAD_SHORT_LINE: CSI Pixel Parser A Pad Short Line This specifies how to

Bit	Reset	Description
		deal with shorter than expected line (the number of bytes received is less than the specified word count) short line is not padded (will output less pixels than expected). This option is not recommended and may cause other modules that receives CSI output stream to hang up. 0 = PAD0S : short line is padded by pixel of zeros such that the expected number of output pixels is correct. Due to the time required to do the padding, subsequent line packet maybe discarded and therefore may cause a short frame (total number of lines per frame is less than expected). 1 = PAD1S : short line is padded by pixel of ones such that the expected number of output pixels is correct. Due to the time required to do the padding, subsequent line packet maybe discarded and therefore may cause a short frame (total number of lines per frame is less than expected). 2 = NOPAD
21:20	X	CSI_PPA_EMBEDDED_DATA_OPTIONS: CSI Pixel Parser A Embedded Data Options. This specifies how to deal with embedded data within the specified input stream assuming that the CSI_PPA_DATA_TYPE is not embedded data and assuming that embedded data is not already processed by other CSI pixel stream processor. Output embedded data as 8-bpp arbitrary data stream. 0 = DISCARD : discard (throw away) embedded data 1 = EMBEDDED
19:16	X	CSI_PPA_OUTPUT_FORMAT_OPTIONS: CSI Pixel Parser A Output Format Options This parameter specifies options for output data format. Output for storing RAW data to memory through ISP. Undefined LS color bits for RGB_666, RGB_565, RGB_555, and RGB_444, will be zeroed. 0 = ARBITRARY : Output as 8-bit arbitrary data stream This may be used for compressed JPEG stream 1 = PIXEL : Output the normal 1 pixel/clock. Undefined LS color bits for RGB_666, RGB_565, RGB_555, and RGB_444, will be zeroed. 2 = PIXEL_REP : Same as PIXEL except MS color bits, for RGB_666, RGB_565, RGB_555, and RGB_444, will be replicated to their undefined LS bits. 3 = STORE
15:14	X	CSI_PPA_VIRTUAL_CHANNEL_ID: CSI Pixel Parser A Virtual Channel Identifier This is CSI compatible virtual channel identifier as defined in CSI specification. If the source stream contains packet headers and CSI_PPA_DATA_IDENTIFIER is ENABLED this value will be compared to the CSI Virtual Channel Identifier value in the 2 MSB of the CSI Data Identifier (DI) byte. This value will be ignored if the source stream doesn't contain packet headers, or CSI_PPA_DATA_IDENTIFIER is DISABLED, then this value will be ignored. 0 = ONE 1 = TWO 2 = THREE 3 = FOUR
13:8	X	CSI_PPA_DATA_TYPE: CSI Pixel Parser A Data Type This is CSI compatible data type as defined in CSI specification. If the source stream contains packet headers this value can be compared to the CSI Data Type value in the 6 LSB of the CSI Data Identifier (DI) byte. If the source stream doesn't contain packet headers, or CSI_PPA_DATA_IDENTIFIER is DISABLED, this value will be used to determine how the stream will be converted to pixels. 24 = YUV420_8 25 = YUV420_10 26 = LEG_YUV420_8 28 = YUV420CSPS_8 29 = YUV420CSPS_10 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

Bit	Reset	Description
		48 = ARB_DT1 49 = ARB_DT2 50 = ARB_DT3 51 = ARB_DT4
7	X	CSI_PPA_CRC_CHECK: CSI Pixel Parser A Data CRC Check This parameter specifies whether the last 2 bytes of packet should be treated as CRC checksum and used to perform CRC check on the payload data. Note that in case there are 2 bytes of data CRC at the end of the packet, the packet word count does not include the CRC bytes. Data CRC Check is enabled. 0 = DISABLE : Data CRC Check is disabled regardless of whether there are CRC checksum at the end of the packet. 1 = ENABLE
6	X	CSI_PPA_WORD_COUNT_SELECT: CSI Pixel Parser A Word Count Select This parameter is effective only if packet header is sent as part of the stream. The number of bytes per line is to be extracted from Word Count field in the packet header. Note that if the serial link is not error free, programming this bit to HEADER may be dangerous because the word count information in the header may be corrupted. 0 = REGISTER : Word Count in packet header is ignored and the number of bytes per line/packet is specified by CSI_PPA_WORD_COUNT. Payload CRC check will not be valid if the word count in CSI_PPA_WORD_COUNT is different than the count in the packet header. It is recommended to always program this bit to REGISTER and always program CSI_PPA_WORD_COUNT. 1 = HEADER
5	X	CSI_PPA_DATA_IDENTIFIER: CSI Pixel Parser A Data Identifier (DI) byte processing This parameter is effective only if packet header is sent as part of the stream. Enabled - Data Identifier byte in packet header should be compared against the CSI_PPA_DATA_TYPE and the CSI_PPA_VIRTUAL_CHANNEL_ID. 0 = DISABLED : Disabled - Data Identifier byte in packet header should be ignored (not checked against CSI_PPA_DATA_TYPE and against CSI_PPA_VIRTUAL_CHANNEL_ID). In this case, CSI_PPA_DATA_TYPE specifies the stream data format. 1 = ENABLED
4	X	CSI_PPA_PACKET_HEADER: CSI Pixel Parser A Packet Header processing This specifies whether packet header is sent in the beginning of packet or not. Packet header is sent. This setting should be used if the stream source is CSI Interface A or B. 0 = NOT_SENT : Packet header is not sent. This setting should not be used if the stream source is CSI Interface A or B. Unless CSI-A, or CSI-B, is operating in a stream capture debug mode. In this case, CSI_PPA_DATA_TYPE specifies the stream data format and the number of bytes per line/packet is specified by CSI_PPA_WORD_COUNT. This implies that a packet footer is also not sent. In this case, no packet footer CRC check should be performed. 1 = SENT
2:0	0x0	CSI_PPA_STREAM_SOURCE: CSI Pixel Parser A Stream Source Host 0 = CSI_A : CSI Interface A 1 = CSI_B : CSI Interface B 6 = VI_PORT : VI port 7 = HOST

28.11.5 CSI_PIXEL_STREAM_A_CONTROL1_0

CSI Pixel Stream A Control 1

Offset: 207h | Read/Write: R/W | Reset: 0b00000000 |

Bit	Reset	Description
7:4	0x0	CSI_PPA_TOP_FIELD_FRAME_MASK: CSI Pixel Parser A Top Field Frame Mask
3:0	0x0	CSI_PPA_TOP_FIELD_FRAME: CSI Pixel Parser A Top Field Frame This parameter specifies the frame number for top field detection for interlaced input video stream. Top Field is indicated when each of the least significant four bits of the frame number that has a one in its mask bit matches the corresponding bit in this parameter. In other words, Top Field is detected when the bitwise AND of ~(CSI_PPA_TOP_FIELD_FRAME ^ <frame

Bit	Reset	Description
		number>) & CSI_PPA_TOP_FIELD_FRAME_MASK is one. Frame Number is taken from the WC field of the Frame Start short packet.

28.11.6 CSI_PIXEL_STREAM_A_WORD_COUNT_0

CSI Pixel Stream A Word Count

Offset: 208h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx |

Bit	Reset	Description
15:0	X	CSI_PPA_WORD_COUNT: CSI Pixel Parser A Word Count. This parameter specifies the number of bytes per line/packet in the case where Word Count field in packet header is not used or where packet header is not sent. This count does not include the additional 2 bytes of CRC checksum if data CRC check is enabled. When the input stream comes from a CSI camera port, this parameter must be programmed when CSI_PPA_PAD_SHORT_LINE is set to either PAD0S or PAD1S, no matter whether CSI_PPA_WORD_COUNT_SELECT is set to REGISTER or HEADER. When the input stream comes from the host path or from the VIP path, and the data mode is PAYLOAD_ONLY, this count must be programmed. Given a line width of N pixels, the programming value of this parameters is as follows **Data format** — **Value** YUV420_8 — N bytes YUV420_10 — N/4*5 bytes LEG_YUV420_8 — N/2*3 bytes YUV422_8 — N*2 bytes YUV422_10 — N/2*5 bytes RGB888 — N*3 bytes RGB666 — N/4*9 bytes RGB565 — N*2 bytes RGB555 — N*2 bytes RGB444 — N*2 bytes RAW6 — N/4*3 bytes RAW7 — N/8*7 bytes RAW8 — N bytes RAW10 — N/4*5 bytes RAW12 — N/2*3 bytes RAW14 — N/4*7 bytes

28.11.7 CSI_PIXEL_STREAM_A_GAP_0

CSI Pixel Stream A Gap

Offset: 209h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	PPA_FRAME_MIN_GAP: Minimum number of viclk cycles from end of frame (Video_control = EF) to start of next frame (Video_control = SF). This parameter is to ensure that minimum V-blank time requirement of VI/ISP is satisfied. This field takes effect only when the frame gap of the input stream is less than the specified value.
15:0	0x0	PPA_LINE_MIN_GAP: Minimum number of viclk cycles from end of previous line (Video_control = EL_DATA) to start of next line (Video_control = SL). This parameter is to ensure that minimum H-blank time requirement of VI/ISP is satisfied. This field takes effect only when the line gap of the input stream is less than the specified value.

28.11.8 CSI_PIXEL_STREAM_PPA_COMMAND_0

CSI Pixel Parser A Command

Offset: 20ah | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxx00

Bit	Reset	Description
15:12	X	CSI_PPA_START_MARKER_FRAME_MAX: CSI Pixel Parser A Start Marker Maximum Start Frame is indicated when Min condition above is met and the least significant four bits of the frame number are less than, or equal to, this value.
11:8	X	CSI_PPA_START_MARKER_FRAME_MIN: CSI Pixel Parser A Start Marker Minimum Start Frame is indicated when Max condition below is met and the least significant four bits of the frame number are greater than, or equal to, this value.
4	X	CSI_PPA_VSYNC_START_MARKER: CSI Pixel Parser A VSYNC Start Marker start of frame is indicated when VSYNC signal is received. When the input stream is from the VIP path and the data mode is PACKET, then this field may be programmed to VSYNC. 0 = FSPKT : Start of frame is indicated when a Frame Start short packet is received with a frame number whose least significant four bits are greater than, or equal to, CSI_PPA_START_MARKER_FRAME_MIN and less than, or equal to, CSI_PPA_START_MARKER_FRAME_MAX. When the input stream is from a CSI port, or from the host path, or from the VIP path and the data mode is PAYLOAD_ONLY, then this field may be programmed to FSPKT. 1 = VSYNC
2	X	CSI_PPA_SINGLE_SHOT: CSI Pixel Parser A Single Shot Mode SW should Clear it along with disabling the CSI_PPA_ENABLE, once a frame is captured 0 = DISABLE 1 = ENABLE
1:0	0x0	CSI_PPA_ENABLE: CSI Pixel Parser A Enable This parameter controls CSI Pixel Parser A to start or stop receiving data. Reset (disable immediately) Enabling the pixel Parser does not enable the corresponding input source to receive data. If Pixel parser is enabled later than the corresponding input source, CSI will keep on rejecting incoming stream, till it encounters a valid SF. 0 = NOP : no operation 1 = ENABLE : enable at the next frame start as specified by the CSI Start Marker 2 = DISABLE : disable after current frame end and before next frame start. 3 = RST

28.11.9 CSI_INPUT_STREAM_B_CONTROL_0

CSI Input Stream B Control

Offset: 20fh | Read/Write: R/W | Reset: 0b0111111xxxxxxxxxxx0xx00

Bit	Reset	Description
22:16	0x3f	CSI_B_SKIP_PACKET_THRESHOLD: CSI-B Skip Packet Threshold This value is compared against the internal FIFO that buffer the input streams. A packet will be skipped (discarded) if the pixel stream processor is busy (probably due to padding process of a short line) and the number of entries in the internal FIFO exceeds this threshold value. Note that each entry in the internal FIFO buffer is four bytes. To turn off this feature, set the value to its maximum value (all ones).
4	0x0	CSI_B_SKIP_PACKET_THRESHOLD_ENABLE: Enables skip packet threshold feature. Skip packet feature is enabled. 0 = DISABLE : Skip packet feature is disabled. 1 = ENABLE
1:0	0x0	CSI_B_DATA_LANE: CSI-B Data Lane 0= 1 data lane 1= 2 data lanes (not supported on SC17 & SC25) 2= 3 data lanes (not supported on SC17 & SC25) 3= 4 data lanes (not supported on SC17 & SC25)

28.11.10 CSI_PIXEL_STREAM_B_CONTROL0_0

CSI Pixel Stream A Control 0

Offset: 211h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000

Bit	Reset	Description
29:28	X	CSI_PPB_PAD_FRAME: CSI Pixel Parser B Pad Frame This specifies how to deal with frames that are shorter (fewer lines) than expected. Short frames are usually caused by line packets being dropped because of packet errors. Expected frame height is specified in PPB_EXP_FRAME_HEIGHT. To do padding the value in CSI_PPB_WORD_COUNT needs to be set to the number of input bytes in each lines payload. Short frames will not be padded out. 0 = PAD0S : Lines of all zeros will be used to pad out frames that are shorter than expected height. PPB_EXP_FRAME_HEIGHT must be programmed to an appropriate value if this fields is set to PAD0S. 1 = PAD1S : Lines of all ones will be used to pad out frames that are shorter than expected height. PPB_EXP_FRAME_HEIGHT must be programmed to an appropriate value if this fields is set to PAD1S. 2 = NOPAD
27	X	CSI_PPB_HEADER_EC_ENABLE: CSI Pixel Parser B Packet Header Error Correction Enable This parameter specifies whether single bit errors in the packet header will be automatically corrected, or not. Single bit errors in the header will not be corrected. Header ECC check will still set header ECC status bits and the packet will be processed by Pixel Parser B. DISABLE should not be used when processing interleaved streams (Same stream going to both PPA and PPB). 0 = ENABLE : Single bit errors in the header will be automatically corrected. 1 = DISABLE
25:24	X	CSI_PPB_PAD_SHORT_LINE: CSI Pixel Parser B Pad Short Line This specifies how to deal with shorter than expected line (the number of bytes received is less than the specified word count) short line is not padded (will output less pixels than expected). This option is not recommended and may cause other modules that receives CSI output stream to hang up. 0 = PAD0S : short line is padded by pixel of zeros such that the expected number of output pixels is correct. Due to the time required to do the padding, subsequent line packet maybe discarded and therefore may cause a short frame (total number of lines per frame is less than expected). 1 = PAD1S : short line is padded by pixel of ones such that the expected number of output pixels is correct. Due to the time required to do the padding, subsequent line packet maybe discarded and therefore may cause a short frame (total number of lines per frame is less than expected). 2 = NOPAD
21:20	X	CSI_PPB_EMBEDDED_DATA_OPTIONS: CSI Pixel Parser B Embedded Data Options This specifies how to deal with embedded data within the specified input stream assuming that the CSI_PPB_DATA_TYPE is not embedded data and assuming that embedded data is not already processed by other CSI pixel stream processor. output embedded data as 8-bpp arbitrary data stream. 0 = DISCARD : discard (throw away) embedded data 1 = EMBEDDED
19:16	X	CSI_PPB_OUTPUT_FORMAT_OPTIONS: CSI Pixel Parser B Output Format Options This parameter specifies output data format. Output for storing RAW data to memory through ISP. Undefined LS color bits for RGB_666, RGB_565, RGB_555, and RGB_444, will be zeroed. 0 = ARBITRARY : Output as 8-bit arbitrary data stream This may be used for compressed JPEG stream 1 = PIXEL : Output the normal 1 pixel/clock. Undefined LS color bits for RGB_666, RGB_565, RGB_555, and RGB_444, will be zeroed. 2 = PIXEL_REP : Same as PIXEL except MS color bits, for RGB_666, RGB_565, RGB_555, and RGB_444, will be replicated to their undefined LS bits. 3 = STORE
15:14	X	CSI_PPB_VIRTUAL_CHANNEL_ID: CSI Pixel Parser B Virtual Channel Identifier This is CSI compatible virtual channel identifier as defined in CSI specification. If the source stream contains packet headers and CSI_PPB_DATA_IDENTIFIER is ENABLED this value will be compared to the CSI Virtual Channel Identifier value in the 2 MSB of the CSI Data Identifier (DI) byte. This value will be ignored if the source stream doesn't contain

Bit	Reset	Description
		packet headers, or CSI_PPB_DATA_IDENTIFIER is DISABLED, then this value will be ignored. 0 = ONE 1 = TWO 2 = THREE 3 = FOUR
13:8	X	CSI_PPB_DATA_TYPE: CSI Pixel Parser B Data Type This is CSI compatible data type as defined in CSI specification. If the source stream contains packet headers this value can be compared to the CSI Data Type value in the 6 LSB of the CSI Data Identifier (DI) byte. If the source stream doesn't contain packet headers, or CSI_PPB_DATA_IDENTIFIER is DISABLED, this value will be used to determine how the stream will be converted to pixels. 24 = YUV420_8 25 = YUV420_10 26 = LEG_YUV420_8 28 = YUV420CSPS_8 29 = YUV420CSPS_10 30 = YUV422_8 31 = YUV422_10 32 = RGB444 33 = RGB555 34 = RGB565 35 = RGB666 36 = RGB888 40 = RAW6 41 = RAW7 42 = RAW8 43 = RAW10 44 = RAW12 45 = RAW14 48 = ARB_DT1 49 = ARB_DT2 50 = ARB_DT3 51 = ARB_DT4
7	X	CSI_PPB_CRC_CHECK: CSI Pixel Parser B Data CRC Check This parameter specifies whether the last 2 bytes of packet should be treated as CRC checksum and used to perform CRC check on the payload data. Note that in case there are 2 bytes of data CRC at the end of the packet, the packet word count does not include the CRC bytes. Data CRC Check is enabled. 0 = DISABLE : Data CRC Check is disabled regardless of whether there are CRC checksum at the end of the packet. 1 = ENABLE
6	X	CSI_PPB_WORD_COUNT_SELECT: CSI Pixel Parser B Word Count Select This parameter is effective only if packet header is sent as part of the stream. The number of bytes per line is to be extracted from Word Count field in the packet header. Note that if the serial link is not error free, programming this bit to HEADER may be dangerous because the word count information in the header may be corrupted. 0 = REGISTER : Word Count in packet header is ignored and the number of bytes per line/packet is specified by CSI_PPB_WORD_COUNT. Payload CRC check will not be valid if the word count in CSI_PPB_WORD_COUNT is different than the count in the packet header. It is recommended to always program this bit to REGISTER and always program CSI_PPB_WORD_COUNT. 1 = HEADER
5	X	CSI_PPB_DATA_IDENTIFIER: CSI Pixel Parser B Data Identifier (DI) byte processing This parameter is effective only if packet header is sent as part of the stream. Enabled - Data Identifier byte in packet header should be compared against the CSI_PPB_DATA_TYPE and the CSI_PPB_VIRTUAL_CHANNEL_ID. 0 = DISABLED : Disabled - Data Identifier byte in packet header should be ignored (not checked against CSI_PPB_DATA_TYPE and against CSI_PPB_VIRTUAL_CHANNEL_ID). In this case, CSI_PPB_DATA_TYPE specifies the stream data format. 1 = ENABLED
4	X	CSI_PPB_PACKET_HEADER: CSI Pixel Parser B Packet Header processing This specifies whether packet header is sent in the beginning of packet or not. Packet header is sent. This setting should be used if the stream source is CSI Interface A or B.

Bit	Reset	Description
		0 = NOT_SENT : Packet header is not sent. This setting should not be used if the stream source is CSI Interface A or B. Unless CSI-A, or CSI-B, is operating in a stream capture debug mode. In this case, CSI_PPB_DATA_TYPE specifies the stream data format and the number of bytes per line/packet is specified by CSI_PPB_WORD_COUNT. This implies that a packet footer is also not sent. In this case, no packet footer CRC check should be performed. 1 = SENT
2:0	0x0	CSI_PPB_STREAM_SOURCE: CSI Pixel Parser B Stream Source Host 0 = CSI_A : CSI Interface A 1 = CSI_B : CSI Interface B 6 = VI_PORT : VI port 7 = HOST

28.11.11 CSI_PIXEL_STREAM_B_CONTROL1_0

CSI Pixel Stream B Control 1

Offset: 212h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7:4	0x0	CSI_PPB_TOP_FIELD_FRAME_MASK: CSI Pixel Parser B Top Field Frame Mask
3:0	0x0	CSI_PPB_TOP_FIELD_FRAME: CSI Pixel Parser B Top Field Frame This parameter specifies the frame number for top field detection for interlaced input video stream. Top Field is indicated when each of the least significant four bits of the frame number that has a one in its mask bit matches the corresponding bit in this parameter. In other words, Top Field is detected when the bitwise AND of ~(CSI_PPB_TOP_FIELD_FRAME ^ <frame number>) & CSI_PPB_TOP_FIELD_FRAME_MASK is one. Frame Number is taken from the WC field of the Frame Start short packet.

28.11.12 CSI_PIXEL_STREAM_B_WORD_COUNT_0

CSI Pixel Stream A Word Count

Offset: 213h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	CSI_PPB_WORD_COUNT: CSI Pixel Parser B Word Count This parameter specifies the number of bytes per line/packet in the case where Word Count field in packet header is not used or where packet header is not sent. This count does not include the additional 2 bytes of CRC checksum if data CRC check is enabled. When the input stream comes from a CSI camera port, this parameter must be programmed when CSI_PPB_PAD_SHORT_LINE is set to either PAD0S or PAD1S, no matter whether CSI_PPB_WORD_COUNT_SELECT is set to REGISTER or HEADER. When the input stream comes from the host path or from the VIP path, and the data mode is PAYLOAD_ONLY, this count must be programmed. Given a line width of N pixels, the programming value of this parameters is as follows data format value YUV420_8 N bytes YUV420_10 N/4*5 bytes LEG_YUV420_8 N/2*3 bytes YUV422_8 N*2 bytes YUV422_10 N/2*5 bytes RGB888 N*3 bytes RGB666 N/4*9 bytes RGB565 N*2 bytes RGB555 N*2 bytes RGB444 N*2 bytes RAW6 N/4*3 bytes RAW7 N/8*7 bytes RAW8 N bytes RAW10 N/4*5 bytes RAW12 N/2*3 bytes RAW14 N/4*7 bytes

28.11.13 CSI_PIXEL_STREAM_B_GAP_0

CSI Pixel Stream B Gap

Offset: 214h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	PPB_FRAME_MIN_GAP: Minimum number of viclk cycles from end of frame (Video_control = EF) to start of next frame (Video_control = SF). This parameter is to ensure that minimum V-blank time requirement of VI/ISP is satisfied. This field takes effect only when the frame gap of the input stream is less than the specified value.
15:0	0x0	PPB_LINE_MIN_GAP: Minimum number of viclk cycles from end of previous line (Video_control = EL_DATA) to start of next line (Video_control = SL). This parameter is to ensure that minimum H-blank time requirement of VI/ISP is satisfied. This field takes effect only when the line gap of the input stream is less than the specified value.

28.11.14 CSI_PIXEL_STREAM_PPB_COMMAND_0

CSI Pixel Parser B Command

Offset: 215h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxx00

Bit	Reset	Description
15:12	X	CSI_PPB_START_MARKER_FRAME_MAX: CSI Pixel Parser B Start Marker Maximum Start Frame is indicated when Min condition above is met and the least significant four bits of the frame number are less than, or equal to, this value.
11:8	X	CSI_PPB_START_MARKER_FRAME_MIN: CSI Pixel Parser B Start Marker Minimum Start Frame is indicated when Max condition below is met and the least significant four bits of the frame number are greater than, or equal to, this value.
4	X	CSI_PPB_VSYNC_START_MARKER: CSI Pixel Parser B VSYNC Start Marker Start of frame is indicated when VSYNC signal is received. When the input stream is from the VIP path and the data mode is PACKET, then this field may be programmed to VSYNC. 0 = FSPKT : Start of frame is indicated when a Frame Start short packet is received with a frame number who's least significant four bits are greater than, or equal to, CSI_PPB_START_MARKER_FRAME_MIN and less than, or equal to, CSI_PPB_START_MARKER_FRAME_MAX. When the input stream is from a CSI port, or from the host path, or from the VIP path and the data mode is PAYLOAD_ONLY, then this field may be programmed to FSPKT. 1 = VSYNC
2	X	CSI_PPB_SINGLE_SHOT: CSI Pixel Parser B Single Shot Mode SW should Clear it alongwith disabling the CSI_PPB_ENABLE, once a frame is captured 0 = DISABLE 1 = ENABLE
1:0	0x0	CSI_PPB_ENABLE: CSI Pixel Parser B Enable This parameter controls CSI Pixel Parser B to start or stop receiving data. reset (disable immediately) Enabling the pixel Parser does not enable the corresponding input source to receive data. If Pixel parser is enabled later than the corresponding input source, csi will keep on rejecting incoming stream, till it encounters a valid SF. 0 = NOP : no operation 1 = ENABLE : enable at the next frame start as specified by the CSI Start Marker 2 = DISABLE : disable after current frame end and before next frame start. 3 = RST

28.11.15 CSI_PHY_CIL_COMMAND_0

CSI Phy and CIL Command

Offset: 21ah | Read/Write: R/W | Reset: 0b00xxxxxxxxxxxxx00

Bit	Reset	Description
17:16	0x0	CSI_B_PHY_CIL_ENABLE: CSI B Phy and CIL Enable This parameter controls CSI B Phy and CIL receiver to start or stop receiving data. disable (reset) 0 = NOP : no operation 1 = ENABLE : enable 2 = DISABLE
1:0	0x0	CSI_A_PHY_CIL_ENABLE: CSI A Phy and CIL Enable This parameter controls CSI A Phy and CIL receiver to start or stop receiving data. disable (reset) 0 = NOP : no operation 1 = ENABLE : enable 2 = DISABLE

28.11.16 CSI_PHY_CILA_CONTROL0_0

CSI-A Phy and CIL Control

Offset: 21bh | Read/Write: R/W | Reset: 0b000010

Bit	Reset	Description
5	0x0	CILA_BYPASS_LP_SEQ: The LP signals should sequence through LP11->LP01->LP00 state, to indicate to CLOCK CIL about the mode switching to HS Rx mode. In case Camera is enabled earlier than CIL , it is highly likely that camera sends this control sequence sooner than CIL can detect it. Enabling this bit allows the CLOCK CIL to overlook the LP control sequence and step in HS Rx mode directly looking at LP00 only.
4	0x0	CILA_SINGLE_SAMPLE: The LP signals are sampled using csi_cil_clk. Normally this happens on 2 clock edges assuming the clock is running at least 50 Mhz. If the clock needs to run slower, then this bit can be SET so that the sampling takes place on a single edge (clock rate is 25 Mhz min). This sampling may not be as reliable so setting this bit is not recommended.
3:0	0x2	CILA_THS_SETTLE: When moving from LP mode to High Speed (LP11->LP01->LP00), this setting determines how many csicil clock cycles (72 MHz lp clock cycles) to wait, after LP00, before starting to look at the data.

28.11.17 CSI_PHY_CILB_CONTROL0_0

CSI-B Phy and CIL Control

Offset: 21ch | Read/Write: R/W | Reset: 0b000010

Bit	Reset	Description
5	0x0	CILB_BYPASS_LP_SEQ: see CILA_BYPASS_LP_SEQ above
4	0x0	CILB_SINGLE_SAMPLE: see CILA_SINGLE_SAMPLE above
3:0	0x2	CILB_THS_SETTLE: When moving from LP mode to High Speed (LP11->LP01->LP00), this setting determines how many csicil clock cycles (72 MHz lp clock cycles) to wait, after LP00, before starting to look at the data.

28.11.18 CSI_CSI_PIXEL_PARSER_STATUS_0

Pixel Parser Status

These status bits are cleared to zero when its bit position is written with one. For example write 0x2 to CSI_PIXEL_PARSER_STATUS will clear only PPA_ILL_WD_CNT.

Offset: 21eh | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	HPH_UNC_HDR_ERR: Uncorrectable Header Error, Set when the Host port header parser parses a header with a multi bit error. This error will be detected by the headers ECC, but can't be corrected. The packet will be discarded.
30	X	HPV_UNC_HDR_ERR: Uncorrectable Header Error, Set when the VI port header parser parses a header with a multi bit error. This error will be detected by the headers ECC, but can't be corrected. The packet will be discarded.
26	X	PPB_SPARE_STATUS_1: PPB Spare Status bit. This bit will get set when Pixel Parser B has a line timeout. Line timeout needs to be enabled by setting PPB_ENABLE_LINE_TIMEOUT and programming PPB_MAX_CLOCKS for the MAX clocks between lines.
25	X	PPB_INTERFRAME_LINE: Set when CSI-PPB receives a request to output a line that is not in the active part of the frame output. That is after EF and before SF, or before start marker is found. The interframe line will not be outputted by the Pixel Parser.
24	X	PPB_EXTRA_SF: Set when CSI-PPB receives a SF when it is expecting an EF. This happens when EF of the frame gets corrupted before arriving CSI. CSI-PPB will insert a fake EF and the drop the current frame with Correct SF.
23	X	PPB_SHORT_FRAME: Set when CSI-PPB receives a short frame. This bit gets set even if CSI_PPB_PAD_FRAME specifies that short frames are to be padded to the correct line length.
22	X	PPB_STMERR: Stream Error, set when the control output of PPB doesn't follow the correct sequence. The correct sequence for CSI is: SF -> (SL_DATA or EF), SL_DATA -> (DATA or EL_DATA), DATA -> EL_DATA, EL_DATA -> (SL_DATA or EF), EF -> SF. Stream Errors can be caused by receiving a corrupted stream, or a CSI RTL bug.
21	X	PPB_FIFO_OVRF: FIFO Overflow, set when the fifo that is feeding packets to PPB overflows.
20	X	PPB_PL_CRC_ERR: PayLoad CRC Error, Set when a packet that was processed by PPB had a payload CRC error.
19	X	PPB_SL_PKT_DROPPED: Short Line Packet Dropped, set when a incoming packet gets dropped because the input FIFO level reaches CSI_B_SKIP_PACKET_THRESHOLD when padding a short line.
18	X	PPB_SL_PROCESSED: Short Line Processed, Set when a line with a payload that is shorter than its packet header word count is processed by PPB.
17	X	PPB_ILL_WD_CNT: Illegal Word Count, set when a line with a word count that doesn't generate an integer number of pixels (Unused bytes at the end of payload) is processed by PPB.
16	X	PPB_HDR_ERR_COR: Header Error Corrected, set when a packet that was processed by PPB has a single bit header error. This error will be detected by the headers ECC, and corrected by it if header error correction is enabled (CSI_B_HEADER_EC_ENABLE = 0). This flag will be set and the packet will be processed even if the error is not corrected.
15	X	HPB_UNC_HDR_ERR: Uncorrectable Header Error, Set when header parser B parses a header with a multi bit error. This error will be detected by the headers ECC, but can't be corrected. The packet will be discarded. a multi bit error. This error will be detected by the headers ECC, but can't be corrected. The packet will be discarded.
14	X	HPA_UNC_HDR_ERR: Uncorrectable Header Error, Set when header parser A parses a header with a multi bit error. This error will be detected by the headers ECC, but can't be corrected. The packet will be discarded.

Bit	Reset	Description
10	X	PPA_SPARE_STATUS_1: PPA Spare Status bit. This bit will get set when Pixel Parser A has a line timeout. Line timeout needs to be enabled by setting PPA_ENABLE_LINE_TIMEOUT and programming PPA_MAX_CLOCKS for the MAX clocks between lines.
9	X	PPA_INTERFRAME_LINE: Set when CSI-PPA receives a request to output a line that is not in the active part of the frame output. That is after EF and before SF, or before start marker is found. The interframe line will not be outputted by the Pixel Parser.
8	X	PPA_EXTRA_SF: Set when CSI-PPA receives a SF when it is expecting an EF. This happens when EF of the frame gets corrupted before arriving CSI. CSI-PPA will insert a fake EF and the drop the current frame with Correct SF.
7	X	PPA_SHORT_FRAME: Set when CSI-PPA receives a short frame. This bit gets set even if CSI_PPA_PAD_FRAME specifies that short frames are to be padded to the correct line length.
6	X	PPA_STMERR: Stream Error, set when the control output of PPA doesn't follow the correct sequence. The correct sequence for CSI is: SF -> (SL_DATA or EF), SL_DATA -> (DATA or EL_DATA), DATA -> EL_DATA, EL_DATA -> (SL_DATA or EF), EF -> SF. Stream Errors can be caused by receiving a corrupted stream, or a CSI RTL bug.
5	X	PPA_FIFO_OVRF: FIFO Overflow, set when the fifo that is feeding packets to PPA overflows.
4	X	PPA_PL_CRC_ERR: PayLoad CRC Error, Set when a packet that was processed by PPA had a payload CRC error.
3	X	PPA_SL_PKT_DROPPED: Short Line Packet Dropped, set when a incoming packet gets dropped because the input FIFO level reaches CSI_A_SKIP_PACKET_THRESHOLD when padding a short line.
2	X	PPA_SL_PROCESSED: Short Line Processed, Set when a line with a payload that is shorter than its packet header word count is processed by PPA.
1	X	PPA_ILL_WD_CNT: Illegal Word Count, set when a line with a word count that doesn't generate an integer number of pixels (Unused bytes at the end of payload) is processed by PPA.
0	X	PPA_HDR_ERR_COR: Header Error Corrected, Set when a packet that was processed by PPA has a single bit header error. This error will be detected by the headers ECC, and corrected by it if header error correction is enabled (CSI_A_HEADER_EC_ENABLE = 0). This flag will be set and the packet will be processed even if the error is not corrected.

28.11.19 CSI_CSI_CIL_STATUS_0

CSI Control and Interface Logic Status

These status bits are cleared to zero when its bit position is written with one. For example write 0x2 to CSI_CIL_STATUS will clear only CILA_SOT_MB_ERR.

Offset: 21fh | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
22	X	CILB_ESC_DATA_REC: Escape Mode Data Received, set when CIL-B receives an Escape Mode Data byte. The Data Byte can be read from bits 23-16 of ESCAPE_MODE_DATA. This status bit will also be cleared when CILB_ESC_CMD_REC is set.
21	X	CILB_ESC_CMD_REC: Escape Mode Command Received, set when CIL-B receives an Escape Mode Command byte. The Command Byte can be read from bits 23-16 of ESCAPE_MODE_COMMAND.
20	X	CILB_CTRL_ERR: Control Error, set when CIL-B detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning into the Escape mode or Turn Around mode (LP00)..
19	X	CILB_ESC_ENTRY_ERR: Escape Mode Entry Error, set when CIL-B detects an Escape

Bit	Reset	Description
		Mode Entry Error. The Escape mode command byte will not be received.
18	X	CILB_SYNC_ESC_ERR: Sync Escape Error, set when CIL-B detects that the wrong (non-multiple of 8) number of bits have been received for an Escape Command, or Data Byte.
17	X	CILB_SOT_MB_ERR: Start of Transmission Multi Bit Error, set when CIL-B detects a multi bit start of transmission byte error in one of the packets SOT bytes. The packet will be discarded.
16	X	CILB_SOT_SB_ERR: Start of Transmission Single Bit Error, set when CIL-B detects a single bit error in one of the packets start of transmission bytes. The packet will be sent to CSI-B for processing.
15	X	MIPI_AUTO_CAL_DONE: MIPI Auto Calibrate done, set when the auto calibrate sequence for MIPI pad bricks is done.
6	X	CILA_ESC_DATA_REC: Escape Mode Data Received, set when CIL-A receives an Escape Mode Data byte. The Data Byte can be read from bits 7-0 of ESCAPE_MODE_DATA. This status bit will also be cleared when CILA_ESC_CMD_REC is set.
5	X	CILA_ESC_CMD_REC: Escape Mode Command Received, set when CIL-A receives an Escape Mode Command byte. The Command Byte can be read from bits 7-0 of ESCAPE_MODE_COMMAND.
4	X	CILA_CTRL_ERR: Control Error, set when CIL-A detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning into the Escape mode or Turn Around mode (LP00).
3	X	CILA_ESC_ENTRY_ERR: Escape Mode Entry Error, set when CIL-A detects an escape mode entry error. The Escape mode command byte will not be received.
2	X	CILA_SYNC_ESC_ERR: Sync Escape Error, set when CIL-A detects that the wrong (non-multiple of 8) number of bits have been received for an Escape Command, or Data Byte.
1	X	CILA_SOT_MB_ERR: Start of Transmission Multi Bit Error, set when CIL-A detects a multi bit start of transmission byte error in one of the packets SOT bytes. The packet will be discarded.
0	X	CILA_SOT_SB_ERR: Start of Transmission Single Bit Error, set when CIL-A detects a single bit error in one of the packets Start of Transmission bytes. The packet will be sent to the CSI-A for processing.

28.11.20 CSI_CSI_PIXEL_PARSER_INTERRUPT_MASK_0

CSI Pixel Parser Interrupt Mask

Offset: 220h | Read/Write: R/W | Reset: 0b00xx00000000000000xx000000000000

Bit	Reset	Description
31	0x0	HPH_UNC_HDR_ERR_INT_MASK: Interrupt Mask for HPH_UNC_HDR_ERR. Generate an interrupt when HPH_UNC_HDR_ERR is set. 0 = DISABLED : Don't generate an interrupt when HPH_UNC_HDR_ERR is set. 1 = ENABLED
30	0x0	HPV_UNC_HDR_ERR_INT_MASK: Interrupt Mask for HPV_UNC_HDR_ERR. Generate an interrupt when HPV_UNC_HDR_ERR is set. 0 = DISABLED : Don't generate an interrupt when HPV_UNC_HDR_ERR is set. 1 = ENABLED
26	0x0	PPB_SPARE_STATUS_1_INT_MASK: Interrupt Mask for PPB_SPARE_STATUS_1. Generate an interrupt when PPB_SPARE_STATUS_1 is set. 0 = DISABLED : Don't generate an interrupt when PPB_SPARE_STATUS_1 is set. 1 = ENABLED
25	0x0	PPB_INTERFRAME_LINE_INT_MASK: Interrupt Mask for PPB_INTERFRAME_LINE. Generate an interrupt when PPB_INTERFRAME_LINE is set. 0 = DISABLED : Don't generate an interrupt when PPB_INTERFRAME_LINE is set. 1 = ENABLED

Bit	Reset	Description
24	0x0	PPB_EXTRA_SF_INT_MASK: Interrupt Mask for PPB_EXTRA_SF. Generate an interrupt when PPB_EXTRA_SF is set. 0 = DISABLED : Don't generate an interrupt when PPB_EXTRA_SF is set. 1 = ENABLED
23	0x0	PPB_SHORT_FRAME_INT_MASK: Interrupt Mask for PPB_SHORT_FRAME. Generate an interrupt when PPB_SHORT_FRAME is set. 0 = DISABLED : Don't generate an interrupt when PPB_SHORT_FRAME is set. 1 = ENABLED
22	0x0	PPB_STMERR_INT_MASK: Interrupt Mask for PPB_STMERR. Generate an interrupt when PPB_STMERR is set. 0 = DISABLED : Don't generate an interrupt when PPB_STMERR is set. 1 = ENABLED
21	0x0	PPB_FIFO_OVRF_INT_MASK: Interrupt Mask for PPB_FIFO_OVRF. Generate an interrupt when PPB_FIFO_OVRF is set. 0 = DISABLED : Don't generate an interrupt when PPB_FIFO_OVRF is set. 1 = ENABLED
20	0x0	PPB_PL_CRC_ERR_INT_MASK: Interrupt Mask for PPB_PL_CRC_ERR. Generate an interrupt when PPB_PL_CRC_ERR is set. 0 = DISABLED : Don't generate an interrupt when PPB_PL_CRC_ERR is set. 1 = ENABLED
19	0x0	PPB_SL_PKT_DROPPED_INT_MASK: Interrupt Mask for PPB_SL_PKT_DROPPED. Generate an interrupt when PPB_SL_PKT_DROPPED is set. 0 = DISABLED : Don't generate an interrupt when PPB_SL_PKT_DROPPED is set. 1 = ENABLED
18	0x0	PPB_SL_PROCESSED_INT_MASK: Interrupt Mask for PPB_SL_PROCESSED. Generate an interrupt when PPB_SL_PROCESSED is set. 0 = DISABLED : Don't generate an interrupt when PPB_SL_PROCESSED is set. 1 = ENABLED
17	0x0	PPB_ILL_WD_CNT_INT_MASK: Interrupt Mask for PPB_ILL_WD_CNT. Generate an interrupt when PPB_ILL_WD_CNT is set. 0 = DISABLED : Don't generate an interrupt when PPB_ILL_WD_CNT is set. 1 = ENABLED
16	0x0	PPB_HDR_ERR_COR_INT_MASK: Interrupt Mask for PPB_HDR_ERR_COR. Generate an interrupt when PPB_HDR_ERR_COR is set. 0 = DISABLED : Don't generate an interrupt when PPB_HDR_ERR_COR is set. 1 = ENABLED
15	0x0	HPB_UNC_HDR_ERR_INT_MASK: Interrupt Mask for HPB_UNC_HDR_ERR. Generate an interrupt when HPB_UNC_HDR_ERR is set. 0 = DISABLED : Don't generate an interrupt when HPB_UNC_HDR_ERR is set. 1 = ENABLED
14	0x0	HPA_UNC_HDR_ERR_INT_MASK: Interrupt Mask for HPA_UNC_HDR_ERR. Generate an interrupt when HPA_UNC_HDR_ERR is set. 0 = DISABLED : Don't generate an interrupt when HPA_UNC_HDR_ERR is set. 1 = ENABLED
10	0x0	PPA_SPARE_STATUS_1_INT_MASK: Interrupt Mask for PPA_SPARE_STATUS_1. Generate an interrupt when PPA_SPARE_STATUS_1 is set. 0 = DISABLED : Don't generate an interrupt when PPA_SPARE_STATUS_1 is set. 1 = ENABLED
9	0x0	PPA_INTERFRAME_LINE_INT_MASK: Interrupt Mask for PPA_INTERFRAME_LINE. Generate an interrupt when PPA_INTERFRAME_LINE is set. 0 = DISABLED : Don't generate an interrupt when PPA_INTERFRAME_LINE is set. 1 = ENABLED
8	0x0	PPA_EXTRA_SF_INT_MASK: Interrupt Mask for PPA_EXTRA_SF. Generate an interrupt when PPA_EXTRA_SF is set. 0 = DISABLED : Don't generate an interrupt when PPA_EXTRA_SF is set. 1 = ENABLED
7	0x0	PPA_SHORT_FRAME_INT_MASK: Interrupt Mask for PPA_SHORT_FRAME. Generate an interrupt when PPA_SHORT_FRAME is set.

Bit	Reset	Description
		0 = DISABLED : Don't generate an interrupt when PPA_SHORT_FRAME is set. 1 = ENABLED
6	0x0	PPA_STMERR_INT_MASK: Interrupt Mask for PPA_STMERR. Generate an interrupt when PPA_STMERR is set. 0 = DISABLED : Don't generate an interrupt when PPA_STMERR is set. 1 = ENABLED
5	0x0	PPA_FIFO_OVRF_INT_MASK: Interrupt Mask for PPA_FIFO_OVRF. Generate an interrupt when PPA_FIFO_OVRF is set. 0 = DISABLED : Don't generate an interrupt when PPA_FIFO_OVRF is set. 1 = ENABLED
4	0x0	PPA_PL_CRC_ERR_INT_MASK: Interrupt Mask for PPA_PL_CRC_ERR. Generate an interrupt when PPA_PL_CRC_ERR is set. 0 = DISABLED : Don't generate an interrupt when PPA_PL_CRC_ERR is set. 1 = ENABLED
3	0x0	PPA_SL_PKT_DROPPED_INT_MASK: Interrupt Mask for PPA_SL_PKT_DROPPED. Generate an interrupt when PPA_SL_PKT_DROPPED is set. 0 = DISABLED : Don't generate an interrupt when PPA_SL_PKT_DROPPED is set. 1 = ENABLED
2	0x0	PPA_SL_PROCESSED_INT_MASK: Interrupt Mask for PPA_SL_PROCESSED. Generate an interrupt when PPA_SL_PROCESSED is set. 0 = DISABLED : Don't generate an interrupt when PPA_SL_PROCESSED is set. 1 = ENABLED
1	0x0	PPA_ILL_WD_CNT_INT_MASK: Interrupt Mask for PPA_ILL_WD_CNT. Generate an interrupt when PPA_ILL_WD_CNT is set. 0 = DISABLED : Don't generate an interrupt when PPA_ILL_WD_CNT is set. 1 = ENABLED
0	0x0	PPA_HDR_ERR_COR_INT_MASK: Interrupt Mask for PPA_HDR_ERR_COR. Generate an interrupt when PPA_HDR_ERR_COR is set. 0 = DISABLED : Don't generate an interrupt when PPA_HDR_ERR_COR is set. 1 = ENABLED

28.11.21 CSI_CSI_CIL_INTERRUPT_MASK_0

CSI Control and Interface Logic Interrupt Mask

Offset: 221h | Read/Write: R/W | Reset: 0b00000000xxxxxx000000000

Bit	Reset	Description
22	0x0	CILB_ESC_DATA_REC_INT_MASK: Interrupt Mask for CILB_ESC_DATA_REC. Generate an interrupt when CILB_ESC_DATA_REC is set. 0 = DISABLED: Don't generate an interrupt when CILB_ESC_DATA_REC is set. 1 = ENABLED
21	0x0	CILB_ESC_CMD_REC_INT_MASK: Interrupt Mask for CILB_ESC_CMD_REC. Generate an interrupt when CILB_ESC_CMD_REC is set. 0 = DISABLED: Don't generate an interrupt when CILB_ESC_CMD_REC is set. 1 = ENABLED
20	0x0	CILB_CTRL_ERR_INT_MASK: Interrupt Mask for CILB_CTRL_ERR. Generate an interrupt when CILB_CTRL_ERR is set. 0 = DISABLED: Don't generate an interrupt when CILB_CTRL_ERR is set. 1 = ENABLED
19	0x0	CILB_ESC_ENTRY_ERR_INT_MASK: Interrupt Mask for CILB_ESC_ENTRY_ERR. Generate an interrupt when CILB_ESC_ENTRY_ERR is set. 0 = DISABLED: Don't generate an interrupt when CILB_ESC_ENTRY_ERR is set. 1 = ENABLED
18	0x0	CILB_SYNC_ESC_ERR_INT_MASK: Interrupt Mask for CILB_SYNC_ESC_ERR. Generate an interrupt when CILB_SYNC_ESC_ERR is set. 0 = DISABLED: Don't generate an interrupt when CILB_SYNC_ESC_ERR is set.

Bit	Reset	Description
		1 = ENABLED
17	0x0	CILB_SOT_MB_ERR_INT_MASK: Interrupt Mask for CILB_SOT_MB_ERR. Generate an interrupt when CILB_SOT_MB_ERR is set. 0 = DISABLED: Don't generate an interrupt when CILB_SOT_MB_ERR is set. 1 = ENABLED
16	0x0	CILB_SOT_SB_ERR_INT_MASK: Interrupt Mask for CILB_SOT_SB_ERR. Generate an interrupt when CILB_SOT_SB_ERR is set. 0 = DISABLED: Don't generate an interrupt when CILB_SOT_SB_ERR is set. 1 = ENABLED
15	0x0	MIPI_AUTO_CAL_DONE_INT_MASK: Interrupt Mask for MIPI_AUTO_CAL_DONE. Generate an interrupt when MIPI_AUTO_CAL_DONE is set. 0 = DISABLED: Don't generate an interrupt when MIPI_AUTO_CAL_DONE is set. 1 = ENABLED
6	0x0	CILA_ESC_DATA_REC_INT_MASK: Interrupt Mask for CILA_ESC_DATA_REC. Generate an interrupt when CILA_ESC_DATA_REC is set. 0 = DISABLED: Don't generate an interrupt when CILA_ESC_DATA_REC is set. 1 = ENABLED
5	0x0	CILA_ESC_CMD_REC_INT_MASK: Interrupt Mask for CILA_ESC_CMD_REC. Generate an interrupt when CILA_ESC_CMD_REC is set. 0 = DISABLED : Don't generate an interrupt when CILA_ESC_CMD_REC is set. 1 = ENABLED
4	0x0	CILA_CTRL_ERR_INT_MASK: Interrupt Mask for CILA_CTRL_ERR. Generate an interrupt when CILA_CTRL_ERR is set. 0 = DISABLED : Don't generate an interrupt when CILA_CTRL_ERR is set. 1 = ENABLED
3	0x0	CILA_ESC_ENTRY_ERR_INT_MASK: Interrupt Mask for CILA_ESC_ENTRY_ERR. Generate an interrupt when CILA_ESC_ENTRY_ERR is set. 0 = DISABLED : Don't generate an interrupt when CILA_ESC_ENTRY_ERR is set. 1 = ENABLED
2	0x0	CILA_SYNC_ESC_ERR_INT_MASK: Interrupt Mask for CILA_SYNC_ESC_ERR. Generate an interrupt when CILA_SYNC_ESC_ERR is set. 0 = DISABLED : Don't generate an interrupt when CILA_SYNC_ESC_ERR is set. 1 = ENABLED
1	0x0	CILA_SOT_MB_ERR_INT_MASK: Interrupt Mask for CILA_SOT_MB_ERR. Generate an interrupt when CILA_SOT_MB_ERR is set. 0 = DISABLED : Don't generate an interrupt when CILA_SOT_MB_ERR is set. 1 = ENABLED
0	0x0	CILA_SOT_SB_ERR_INT_MASK: Interrupt Mask for CILA_SOT_SB_ERR. Generate an interrupt when CILA_SOT_SB_ERR is set. 0 = DISABLED : Don't generate an interrupt when CILA_SOT_SB_ERR is set. 1 = ENABLED

28.11.22 CSI_CSI_READONLY_STATUS_0

CSI Read Only Status

This register is used to return CSI read only status.

Offset: 222h | Read/Write: RO | Reset: 0bxx

Bit	Reset	Description
1	X	CSI_PPB_ACTIVE: One only when Pixel Parser B is capturing frame data.
0	X	CSI_PPA_ACTIVE: One only when Pixel Parser A is capturing frame data.

28.11.23 CSI_ESCAPE_MODE_COMMAND_0

Escape Mode Command

This register is used to receive escape mode command bytes from CIL-A and CIL-B.

Offset: 223h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	CILB_ESC_CMD_BYTE: CIL-B Escape Mode Command Byte, this is the 8 bit entry command that was received, by CIL-B, during the last escape Mode sequence. This command byte can only be assumed to be valid when CILB_ESC_CMD_REC status bit is set.
7:0	X	CILA_ESC_CMD_BYTE: CIL-A Escape Mode Command Byte, this is the 8 bit entry command that was received, by CIL-A, during the last escape Mode sequence. CIL-A monitors Byte Lane 0, only, for escape mode sequences. This command byte can only be assumed to be valid when CILA_ESC_CMD_REC status bit is set.

28.11.24 CSI_ESCAPE_MODE_DATA_0

Escape Mode Data

This register is used to receive escape mode data bytes from CIL-A and CIL-B.

Offset: 224h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	CILB_ESC_DATA_BYTE: CIL-B Escape Mode Data Byte, when read this field returns the last Escape Mode Data byte that was received by CIL-B. Escape Mode Data bytes are the bytes that are received in Escape Mode after receiving the Escape Mode Command. These bytes can be used to implement MIPI's CSI Specs Low Power Data Transmission. This field is only valid when the status bit, CILB_ESC_DATA_REC, is set, and will be overwritten by the next Escape Mode data byte if not read before the next byte come in.
7:0	X	CILA_ESC_DATA_BYTE: CIL-A Escape Mode Data Byte, when read this field returns the last Escape Mode Data byte that was received by CIL-A. Escape Mode Data bytes are the bytes that are received in Escape Mode after receiving the Escape Mode Command. These bytes can be used to implement MIPI's CSI Specs Low Power Data Transmission. This field is only valid when the status bit, CILA_ESC_DATA_REC, is set, and will be overwritten by the next Escape Mode data byte if not read before the next byte come in.

28.11.25 CSI_CILA_PAD_CONFIG0_0

CIL-A Pad Configuration 0

Offset: 225h | Read/Write: R/W | Reset: 0b000x0000000xxx0x000x000x0000111

Bit	Reset	Description
30:28	0x0	PAD_CILA_SLEWDNADJ: Pull down slew rate adjust, default 000 From 000 -> 011, slew rate increases 100 is the same as 000 From 100->111, skew rate decreases.
26:24	0x0	PAD_CILA_SLEWUPADJ: Pull up slew rate adjust, default 000 From 000 -> 011, slew rate increases 100 is the same as 000 From 100->111, skew rate decreases.
23:22	0x0	PAD_CILA_LPDNADJ: Driver pull down impedance control 00 -> 130ohm, default 01 -> 110ohm 10 -> 130ohm, same as 00 11 -> 150ohm
21:20	0x0	PAD_CILA_LPUPADJ: Driver pull up impedance control 00 -> 130ohm, default 01 -> 110ohm 10 -> 130ohm, same as 00 11 -> 150ohm
16	0x0	PAD_CILA_BANDWD_IN: Increase bandwidth of differential receiver
14:12	0x0	PAD_CILA_INADJ1: bit 1 input delay trimmer, each tap delays 20ps

Bit	Reset	Description
10:8	0x0	PAD_CILA_INADJ0: bit 0 input delay trimmer, each tap delays 20ps
6:4	0x0	PAD_CILA_INADJCLK: Clock bit input delay trimmer, each tap delays 20ps
3	0x0	PAD_CILA_PREEMP_EN: HS driver preemphasis enable,1= preemphasis enabled
2	0x1	PAD_CILA_PDIO_CLK: Power down for clock bit, including drivers, receivers and contention detectors
1:0	0x3	PAD_CILA_PDIO: Power down for each data bit, including drivers, receivers and contention detectors

28.11.26 CSI_CILA_PAD_CONFIG1_0

CIL-A Pad Configuration 4

Offset: 226h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx0000000000000000

Bit	R/W	Reset	Description
31:16	RO	X	PAD_CILA_SPARE_RO: Spare Read only bits for CILA Config
15:0	RW	0x0	PAD_CILA_SPARE: Spare bits for CILA Config PAD_CILA_SPARE[15] is being used to disable the CSI-A RTL code that blocks fifo push that are past the end of the line packet. 0: disabled, 1: push blocking enabled

28.11.27 CSI_CILB_PAD_CONFIG0_0

CIL-B Pad Configuration 0

Offset: 227h | Read/Write: R/W | Reset: 0b000x0000000xxx0xxxxx000x00001x1

Bit	Reset	Description
30:28	0x0	PAD_CILB_SLEWDNADJ: Pull down slew rate adjust, default 000 From 000 -> 011, slew rate increases 100 is the same as 000 From 100->111, skew rate decreases.
26:24	0x0	PAD_CILB_SLEWUPADJ: Pull up slew rate adjust, default 000 From 000 -> 011, slew rate increases 100 is the same as 000 From 100->111, skew rate decreases.
23:22	0x0	PAD_CILB_LPDNADJ: Driver pull down impedance control 00 -> 130ohm, default 01 -> 110ohm 10 -> 130ohm, same as 00 11 -> 150ohm
21:20	0x0	PAD_CILB_LPUPADJ: Driver pull up impedance control 00 -> 130ohm, default 01 -> 110ohm 10 -> 130ohm, same as 00 11 -> 150ohm
16	0x0	PAD_CILB_BANDWD_IN: Increase bandwidth of differential receiver
10:8	0x0	PAD_CILB_INADJ0: bit 0 input delay trimmer, each tap delays 20ps
6:4	0x0	PAD_CILB_INADJCLK: Clock bit input delay trimmer, each tap delays 20ps
3	0x0	PAD_CILB_PREEMP_EN: HS driver preemphasis enable,1= preemphasis enabled
2	0x1	PAD_CILB_PDIO_CLK: Power down for clock bit, including drivers, receivers and contention detectors
0	0x1	PAD_CILB_PDIO: Power down for each data bit, including drivers, receivers and contention detectors

28.11.28 CSI_CILB_PAD_CONFIG1_0

CIL-B Pad Configuration 4

Offset: 228h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx0000000000000000

Bit	R/W	Reset	Description
31:16	RO	X	PAD_CILB_SPARE_RO: Spare Read only bits for CILB Config
15:0	RW	0x0	PAD_CILB_SPARE: Spare bits for CILB Config PAD_CILB_SPARE[15] is being used to disable the CSI-B RTL code that blocks fifo push that are past the end of the line packet. 0: disabled, 1: push blocking enabled

28.11.29 CSI_CIL_PAD_CONFIG0_0

CIL Pad Configuration 0

Offset: 229h | Read/Write: R/W | Reset: 0b00000000x000xx10

Bit	Reset	Description
15:8	0x0	PAD_CIL_SPARE: Spare bit for CIL BIAS Config PAD_CIL_SPARE[7] is used is being used to flush VI's Y-FIFO when it is being use as a stream source for one of the Pixel Parsers. Setting PAD_CIL_SPARE[7] to 1 will hold vi2csi_host_stall low. Which will force VI's Y-FIFO to be purged. PAD_CIL_SPARE[7] must be low for the pixel parser to receive source data from VI's Y-FIFO.
6:4	0x0	PAD_CIL_VADJ: VAUXP level adjustment 00 -> no adjustment, default 01 -> 105% 10 -> 110% 11 -> 115% 100 -> no adjustment 101 -> 95% 110 -> 90% 111 -> 85%
1	0x1	PAD_CIL_PDVREG: Power down voltage regulator, 1=power down
0	0x0	PAD_CIL_VBYPASS: Bypass bang gap voltage reference

28.11.30 CSI_CILA_MIPI_CAL_CONFIG_0

Calibration Settings for CIL-A MIPI pads

Offset: 22ah | Read/Write: R/W | Reset: 0b00101010xx100000xxx00000xxx00000

Bit	R/W	Reset	Description
31	RO	0x0	MIPI_CAL_STARTCAL: Writing a one to this bit starts the Calibration State machine. This bit must be set even if both overrides set in order to latch in the over ride value
30	RW	0x0	MIPI_CAL_OVERIDEA: When 0 (normal operation), use the above registers as an offset to the Calibration State machine setting for channel A TERMADJ/HSPUADJ/HSPDADJ values to the Mipi Pads. When 1, use the register values above as the actual value going to channel A TERMADJ/HSPUADJ/HSPDADJ on the Mipi Pads.
29:26	RW	0xa	MIPI_CAL_NOISE_FLT: The DRIVRY & TERMRY signals coming from MIPI Pads are utilized by Calibration state machine for PAD Calibration. The drivry/termry comes from a noisy analog source and it could have some glitches. The filter in calibsm is sensitive to these noises. If the calibration done status does not show up, we can change the sensitivity of the filter through these bits. Ideally this has to be programmed in a range from 10 to 15. For the case when MIPI_CAL_PRESCALE = 2'b00, this needs to be programmed between 2 to 5.
25:24	RW	0x2	MIPI_CAL_PRESCALE: Auto Cal calibration step prescale: Set to 00 when calibration step should be 0.1 us Set to 01 when calibration step should be 0.5 us Set to 10 when calibration step should be 1.0 us Set to 11 when calibration step should be 1.5 us this will keep the mipi bias cal step between 0.1-1.5 usec Default set for 1.0 us calibration step.
21	RW	0x1	MIPI_CAL_SELA: Select the CSIA PADS for auto calibration.
20:16	RW	0x0	MIPI_CAL_HSPDOSA: 2's complement offset for HSPDADJ going to channel A

Bit	R/W	Reset	Description
12:8	RW	0x0	MIPI_CAL_HSPUOSA: 2's complement offset for HSPUADJ going to channel A
4:0	RW	0x0	MIPI_CAL_TERMOSA: 2's complement offset for TERMADJ going to channel A

28.11.31 CSI_CILB_MIPI_CAL_CONFIG_0

Calibration Settings for CIL-B MIPI pads

Offset: 22bh | Read/Write: R/W | Reset: 0b0xxxxxxxx100000xxx00000xxx00000

Bit	Reset	Description
30	0x0	MIPI_CAL_OVERIDEB: When 0 (normal operation), use the above registers as an offset to the Calibration State machine setting for channel B TERMADJ/HSPUADJ/HSPDADJ values to the Mipi Pads. When 1, use the register values above as the actual value going to channel B TERMADJ/HSPUADJ/HSPDADJ on the Mipi Pads. Writing a one to Bit 31 of CILA_MIPI_CAL_CONFIG (MIPI_CAL_STARTCAL) starts the Calibration State machine.
21	0x1	MIPI_CAL_SELB: Select the CSIB PADS for auto calibration.
20:16	0x0	MIPI_CAL_HSPDOSB: 2's complement offset for HSPDADJ going to channel B
12:8	0x0	MIPI_CAL_HSPUOSB: 2's complement offset for HSPUADJ going to channel B
4:0	0x0	MIPI_CAL_TERMOSB: 2's complement offset for TERMADJ going to channel B

28.11.32 CSI_CIL_MIPI_CAL_STATUS_0

CIL MIPI Calibrate Status

Offset: 22ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxx

Bit	Reset	Description
11:8	X	MIPI_CAL_DRIVADJ: Driver code generated by MIPI auto Calibrate. Valid only after auto calibrate sequence has completed (MIPI_CAL_ACTIVE == 0).
7:4	X	MIPI_CAL_TERMADJ: Termination code generated by MIPI auto Calibrate. Valid only after auto calibrate sequence has completed (MIPI_CAL_ACTIVE == 0).
0	X	MIPI_CAL_ACTIVE: One when auto calibrate is active.

28.11.33 CSI_DSI_MIPI_CAL_CONFIG_0

Calibration Settings for DSI MIPI pad

Offset: 234h | Read/Write: R/W | Reset: 0b0xxxxxxxx100000xxx00000xxx00000

Bit	Reset	Description
30	0x0	MIPI_CAL_OVERIDED: When 0 (normal operation), use the above registers as an offset to the Calibration State machine setting for TERMADJ/HSPUADJ/HSPDADJ values to the Mipi Pads. When 1, use the register values above as the actual value going to TERMADJ/HSPUADJ/HSPDADJ on the Mipi Pads. Writing a one to Bit 31 of CILA_MIPI_CAL_CONFIG (MIPI_CAL_STARTCAL) starts the Calibration State machine.
21	0x1	MIPI_CAL_SELD: Select the DSI PADS for auto calibration.
20:16	0x0	MIPI_CAL_HSPDOSD: 2's complement offset for HSPDADJ
12:8	0x0	MIPI_CAL_HSPUOSD: 2's complement offset for HSPUADJ
4:0	0x0	MIPI_CAL_TERMOSD: 2's complement offset for TERMADJ

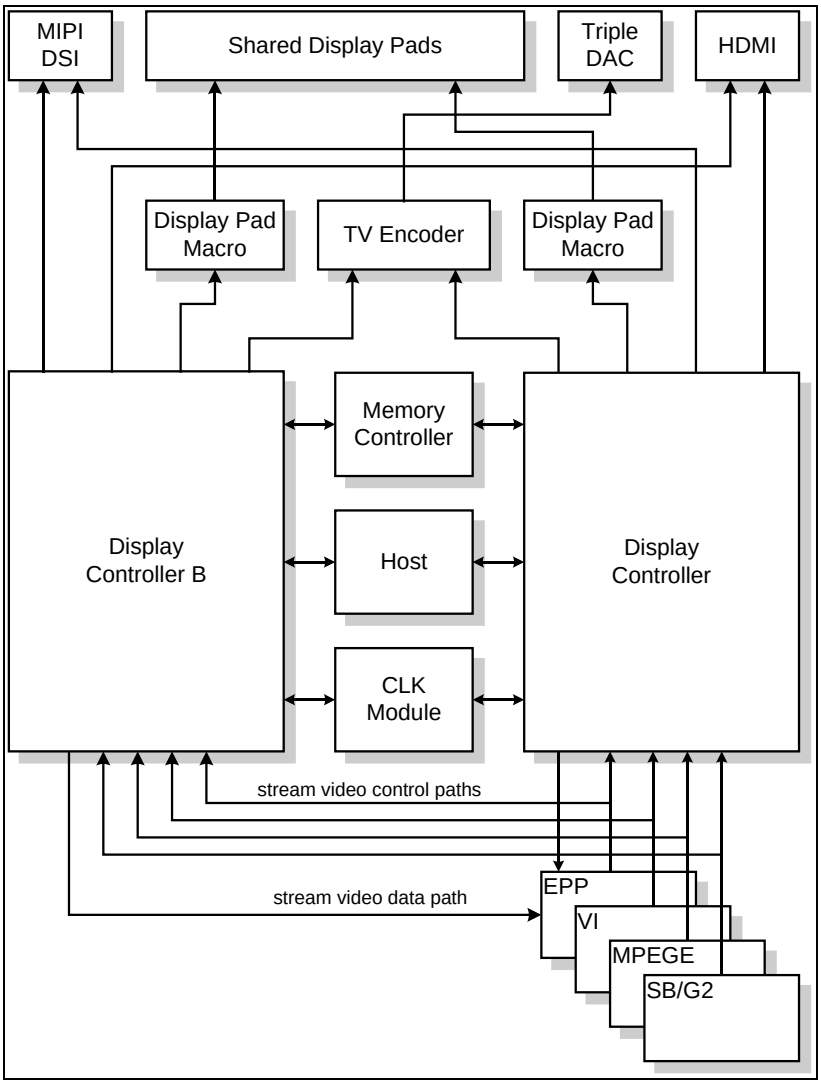
29.0 DISPLAY CONTROLLER

The display controller module interfaces to an external display device, which can be a parallel interface or SPI LCD, a PAL or NTSC TV, a component video HDTV, an HDMI® HDTV, and RGB monitor or a MIPI DSI LCD. It reads rendered graphics or video frame buffers in memory, blends them and sends them to the display. Direct interface is supported directly to most LCD displays with TFT or TFT-like interface.

There are two independent display controllers, so the module described here is instantiated twice.

Each of the two display controllers shares access to the various output resources. There is only one instance of the MIPI DSI, HDMI and TV outputs. Only one display controller may access one of these outputs at any given time.

Figure 86. Display Controller



Display maximum performance is ultimately determined by the availability of memory bandwidth. The line buffer for the DSI transmitter is 1920 bytes in size. This is enough to hold 640 pixels with a pixel depth of 24-bits per pixel, or 960 pixels with a pixel depth of 16-bits per pixel. This will limit the maximum burst rate on the DSI interface for a given pixel clock frequency. However, the buffer has been sized to cope with most reasonable combinations of screen resolution, pixel clock and burst rate.

The line buffer for error diffusion dithering is limited to 1280 pixels per line. This limits the maximum horizontal active area size to 1280 pixels when error diffusion dithering is enabled. For larger displays either full 24-bit color pixel resolution output is required or ordered dithering may be used since it does not require a line buffer.

> **Note:** Display maximum resolution is not limited by counter resolution. The display raster timing generator supports 12-bit counter resolution.

Features

- Three display windows (1 graphics window A and 2 overlay windows B and C)
- Hardware surface blending
- Hardware cursor
- Fully programmable display timing and resolution
- 'Hysteresis' latency FIFO to minimize DRAM power for display refresh in an idle system
- Graceful display underflow handling

TV Output Features

- Supports NTSC, PAL or SECAM color standards.
- Supports standard definition and high definition standard.
- Supports interlaced and progressive scan standard (480i, 576i, 480p, 576p, 1080i, 720p).
- RGB, component YUV/YPbPr, S-video and composite outputs available.
- Programmable controls including sub-carrier frequency, phase, Vsync, number of lines, chroma gain and many others to support all different variations of NTSC and PAL.
- Dual chroma path to enable different bandwidths for the composite path and the component path.
- Notch filter with programmable notch frequencies and band-reject width.
- Full Macrovision™ 7.1L1 encoding for DVD compatibility, which includes automatic AGC cycling.
- WSS (Wide Screen Signaling) support.
- CGMS (Copy Generation Management System) support.
- Closed Captioning.
- Teletext System 625 A (Antiope), B (WST625), C.
- Teletext System 525 B (WST525), C (NABTS), D.
- All the vertical blanking signals (WSS, CGMS, CC, TTX) are fully programmable including line and field selection, edge rise and fall times.
- All sync and Macrovision edges are raised cosine edges.

> **Note:** Although interlaced output is supported, the HDMI® compliance rules do not allow the use of these modes when an HDMI output is also present, as they are **not** supported on the HDMI output.

Clocking

The Display source clock can be selected from a number of sources. The SHIFT_CLK_DIVIDER register value divides the source clock to generate dcsclk, the shift clock. PIXEL_CLK_DIVIDER register value further divides down the shift to generate dcpclk, pixel clock. One pixel is processed every pixel clock and PIXEL_CLK_DIVIDER is programmed such that the correct number of shift clocks per pixel clock is available for the pixel data to be transferred to the LCD.

Display has first, second, and third level clock gating. First level clock gating is controlled by a host register bit. Display has extensive second level clock gating controlled by hardware, and based on activity.

When TVO is enabled, the TVO controls the frame rate of the Display Controller by stalling the Display Controller. TVO sends a stall signal to the Display Controller that qualifies the pixel clock enable, essentially stopping Display Controller's processing until TVO is ready for more data.

29.1 Hardware Interface

29.1.1 Display Pin Definition

Table 93: LCD Display Signals

Name	Description	Type
LCD_CS0_N	LCD Chip Select 0 (Main Display)	Output
LCD_CS1_N	LCD Chip Select 1(Secondary Display)	Output
LCD_D[23:0]	LCD Data	Bidirectional
LCD_DC0	LCD Serial Data/Command or Tearing Input	Bidirectional
LCD_DC1	LCD Data/Command for secondary display	Output
LCD_DE	LCD Data Enable or Tearing Input	Output
LCD_HSYNC	LCD Horizontal Sync	Bidirectional
LCD_M1	LCD Modulation	Output
LCD_PCLK	Main display RGB pixel clock or CPU write enable	Output
LCD_PWR[2:0]	LCD Power Controls	Output
LCD_SCK	Main Display LCD Serial Clock	Output
LCD_SDIN	Main Display LCD Serial Data In or Tearing Input	Input
LCD_SDOUT	Main Display LCD Serial Data Out	Output
LCD_VSYNC	LCD Vertical Sync	Output
LCD_WR_N	Secondary Display Parallel CPU Write Enable, or Serial CPU Clock	Output

Table 94: HDMI Display Signals

Name	Description	Type
HDMI_INT_N	Interrupt	Input
HDMI_PROBE	Reserved	Test
HDMI_RSET	Reference Set. Line to a current set resistor.	Analog
HDMI_TXCN	Transmit Clock –	Output
HDMI_TXCP	Transmit Clock +	Output
HDMI_TXD0N	Transmit (data) Lane 0 –	Output
HDMI_TXD0P	Transmit (data) Lane 0 +	Output
HDMI_TXD1N	Transmit (data) Lane 1 –	Output
HDMI_TXD1P	Transmit (data) Lane 1 +	Output

Name	Description	Type
HDMI_TXD2N	Transmit (data) Lane 2 –	Output
HDMI_TXD2P	Transmit (data) Lane 2 +	Output

Table 95: TV Output CRT Display Signals

Name	Description	Type
CRT_HSYNC	Horizontal Sync	Output
CRT_VSYNC	Vertical Sync	Output
VDAC_B	Output, Blue	Analog Output
VDAC_G	Output, Green	Analog Output
VDAC_R	Output, Red	Analog Output
VDAC_RSET	Resistor Control Input	Analog
VDAC_VREF	Voltage reference input to DAC or voltage reference output	Analog

Table 96: DSI Display Signals

Name	Description	Type
DSI_CLKAN	Clock -	Output
DSI_CLKAP	Clock +	Output
DSI_D1AN	Data Lane 1 -	Output
DSI_D1AP	Data Lane 1 +	Output
DSI_D2AN	Data Lane 2 -	Output
DSI_D2AP	Data Lane 2 +	Output

29.1.2 Interface to Display

Table 97: Display Signal Interface

Pin		Primary							Secondary		
	24-bit RGB 5-wire SPI 1 clk/pixel	18-bit RGB 5-wire SPI 1 clk/pixel	16-bit RGB 5-wire SPI 1 clk/pixel	24-bit CPU 1 clk/pixel	18-bit CPU 1 clk/pixel	16-bit CPU 1 clk/pixel	9-bit CPU 2 clk/pixel 18-bpp	8-bit CPU 2 clk/pixel 16-bpp	5-wire SPI	9-bit CPU 2 clk/pixel 18-bpp	8-bit CPU 2 clk/pixel 16-bpp
LCD_D23	R1			D23						D3	D3
LCD_D22	R0			D22						D2	D2
LCD_D21	G1			D21						D7	D7
LCD_D20	G0			D20						D6	D6
LCD_D19	B1			D19							
LCD_D18	B0			D18						D0	D0
LCD_D17	R7	R5	R4	D17	D17						
LCD_D16	R6	R4	R3	D16	D16						
LCD_D15	R5	R3	R2	D15	D15	D15					
LCD_D14	R4	R2	R1	D14	D14	D14					
LCD_D13	R3	R1	R0	D13	D13	D13					
LCD_D12	R2	R0		D12	D12	D12					
LCD_D11	G7	G5	G5	D11	D11	D11					
LCD_D10	G6	G4	G4	D10	D10	D10					
LCD_D9	G5	G3	G3	D9	D9	D9					
LCD_D8	G4	G2	G2	D8	D8	D8	D8				
LCD_D7	G3	G1	G1	D7	D7	D7	D7	D7			
LCD_D6	G2	G0	G0	D6	D6	D6	D6	D6			
LCD_D5	B7	B5	B4	D5	D5	D5	D5	D5			
LCD_D4	B6	B4	B3	D4	D4	D4	D4	D4			
LCD_D3	B5	B3	B2	D3	D3	D3	D3	D3			
LCD_D2	B4	B2	B1	D2	D2	D2	D2	D2			
LCD_D1	B3	B1	B0	D1	D1	D1	D1	D1			
LCD_D0	B2	B0		D0	D0	D0	D0	D0			
LCD_PWR0									SPI_SDOUT	D4	D4
LCD_PWR1										D8	
LCD_PWR2									SPI_DIN	D5	D5
LCD_PCLK	PCLK	PCLK	PCLK	WR_	WR_	WR_	WR_	WR_			
LCD_WR_									SPI_SCK	WR_	WR_
LCD_VSYNC	VSYNC	VSYNC	VSYNC								
LCD_HSYNC	HSYNC	HSYNC	HSYNC								
LCD_DC1									DC	DC	DC
LCD_CS1_									SPI_CS_	CPU_CS_	CPU_CS_
LCD_M1										D1	D1
LCD_SCK	SPI_SCK	SPI_SCK	SPI_SCK								
LCD_SDOUT	SPI_SDOUT	SPI_SDOUT	SPI_SDOUT								
LCD_SDIN	SPI_SDIN	SPI_SDIN	SPI_SDIN								
LCD_CS0_	SPI_CS_	SPI_CS_	SPI_CS_	CPU_CS_	CPU_CS_	CPU_CS_	CPU_CS_	CPU_CS_			
LCD_DC0	DC	DC	DC	DC	DC	DC	DC	DC			
LCD_DE	DE	DE	DE	Tearing	Tearing	Tearing	Tearing	Tearing			

Note: For CPU LCDs, refer to the color data mapping below. RGB LCDs may or may not include SPI interfaces used for programming

Table 98: CPU Interface Color Mapping

Pin	24-bit, 1 clk/pixel		18-bit, 1 clk/pixel		16-bit, 1 clk/pixel		9-bit, 2 clk/pixel			8-bit, 2 clk/pixel		
	LCD Pin	LCD Color Data	LCD Pin	LCD Color Data	LCD Pin	LCD Color Data	LCD Pin	LCD Color Data Clock 1	LCD Color Data Clock 2	LCD Pin	LCD Color Data Clock 1	LCD Color Data Clock 2
LCD_D23	D23	R7										
LCD_D22	D22	R6										
LCD_D21	D21	R5										
LCD_D20	D20	R4										
LCD_D19	D19	R3										
LCD_D18	D18	R2										
LCD_D17	D17	R1	D17	R5								
LCD_D16	D16	R0	D16	R4								
LCD_D15	D15	G7	D15	R3	D15	R5						
LCD_D14	D14	G6	D14	R2	D14	R4						
LCD_D13	D13	G5	D13	R1	D13	R3						
LCD_D12	D12	G4	D12	R0	D12	R2						
LCD_D11	D11	G3	D11	G5	D11	R1						
LCD_D10	D10	G2	D10	G4	D10	G5						
LCD_D9	D9	G1	D9	G3	D9	G4						
LCD_D8	D8	G0	D8	G2	D8	G3	D8	R5	G2			
LCD_D7	D7	B7	D7	G1	D7	G2	D7	R4	G1	D7	R5	G2
LCD_D6	D6	B6	D6	G0	D6	G1	D6	R3	G0	D6	R4	G1
LCD_D5	D5	B5	D5	B5	D5	G0	D5	R2	B5	D5	R3	G0
LCD_D4	D4	B4	D4	B4	D4	B5	D4	R1	B4	D4	R2	B5
LCD_D3	D3	B3	D3	B3	D3	B4	D3	R0	B3	D3	R1	B4
LCD_D2	D2	B2	D2	B2	D2	B3	D2	G5	B2	D2	G5	B3
LCD_D1	D1	B1	D1	B1	D1	B2	D1	G4	B1	D1	G4	B2
LCD_D0	D0	B0	D0	B0	D0	B1	D0	G3	B0	D0	G3	B1

29.1.3 Interface to Host

The display controller interfaces to the host with a similar architecture as other modules. Display's host interface contains a read FIFO and a write FIFO, allowing an asynchronous boundary between host clock and display clock. Display registers are synchronously written and read in display's clock domain. Class and register writes are treated identically.

Display controller supports several different sync points, allowing software to synchronize events properly. Refer to the display spec files for the sync points that are supported.

Display controller supports several interrupts that can be used for synchronization or for notifying the host of corruption (underflow/overflow). All interrupts are programmable to be level or edge sensitive. Refer to the display spec files for the interrupts that are supported.

29.1.4 Interface to Memory

Display has 5 memory clients: one block read client for window A (128-bit x 64 deep read FIFO), two block read clients for window B (128-bit x 64-deep read FIFO), one block read client for window C (128-bit x 64 deep read FIFO), and one single read client for the cursor. For display controller purposes, the block read client is used as a line read since display controller

needs data on a line by line basis to do vertical scaling. In addition, the read memory clients for window B and C support reading YUV data.

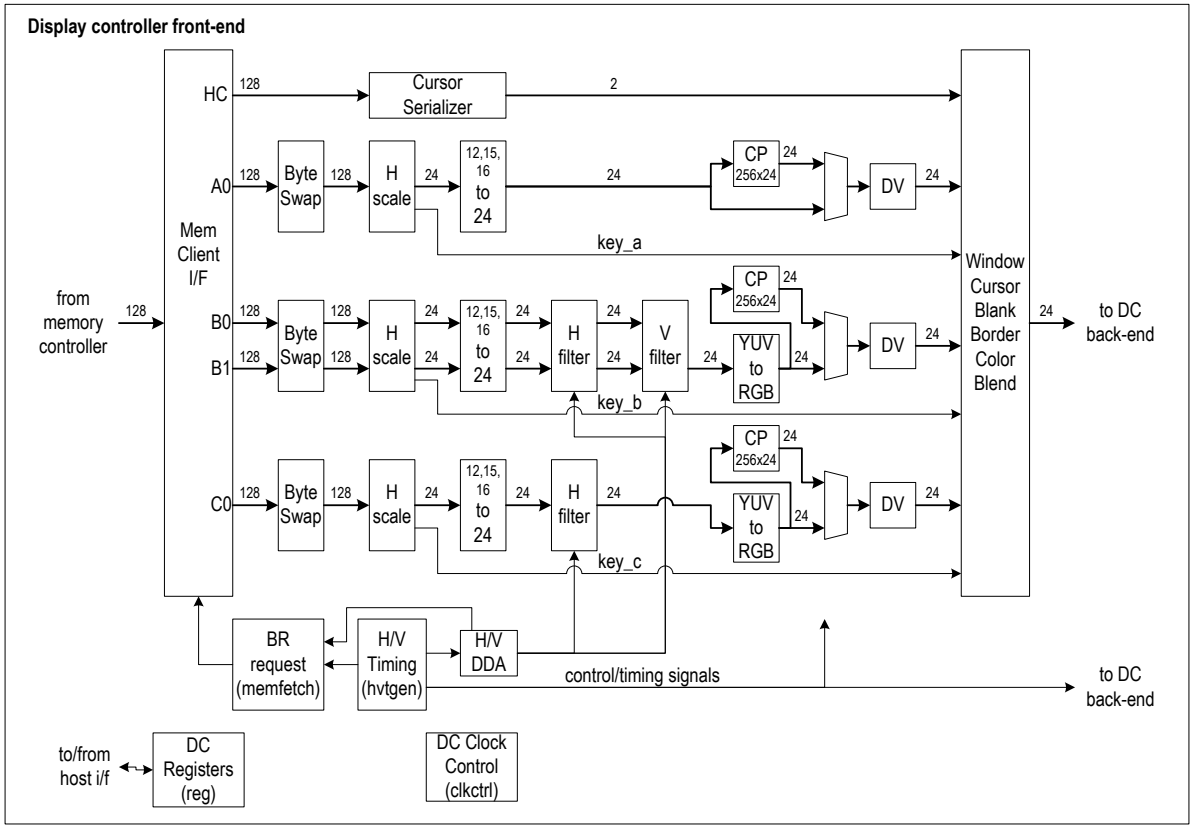
Display controller requires a constant bandwidth and can withstand only a limited latency. For this reason, display controller has the highest priority for memory controller requests. An underflow in memory data will result in a visual corruption on the LCD panel.

29.2 Functionality

29.2.1 Block Diagram

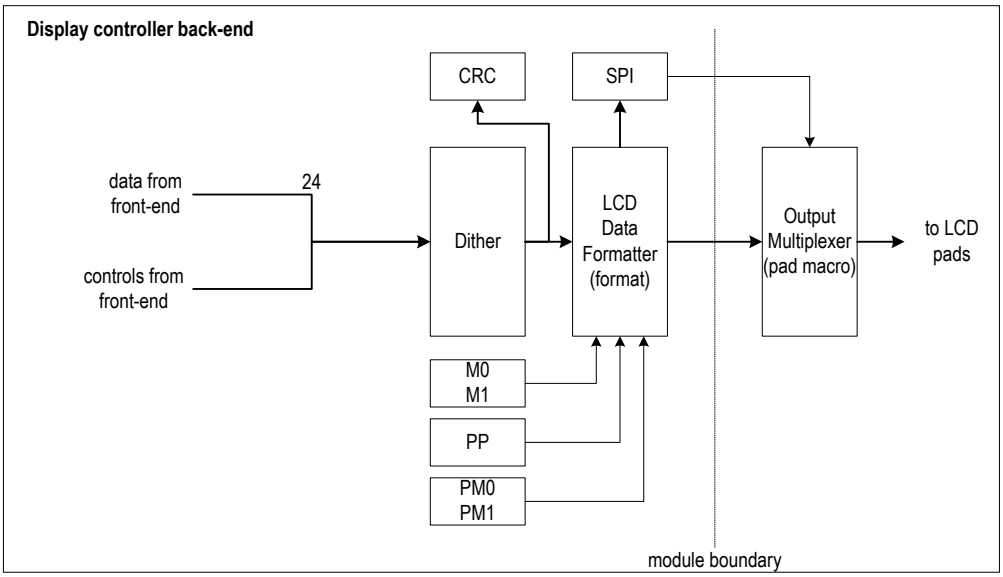
The following shows a block diagram of the display controller module front-end:

Figure 87. Display Controller Front-End Block Diagram



The following shows a block diagram of the display controller module back-end:

Figure 88. Display Controller Back-end Block Diagram



29.2.2 Color Key and Overlay Blend

When more than one active window is overlapping, the color key multiplexer selects between blended window data and pure window data. There are 3 windows (A, B, C) and therefore there are potentially 3 regions (A, B, C) where there is no window overlap and 4 regions (AB, AC, BC, ABC) where there is window overlap.

Blended data is the sum of the weighted active window data. The weight is determined from register programming and is based on a color key match, a fixed weight, an alpha weight, or a weight dependent on the other active window(s). There are separate registers defined for each possible overlap condition.

Color key can be defined in any of the overlapping window. Typically color key is enabled in one of the overlapping windows only. If color key is enabled in more than one of the overlapping windows then the color key multiplexer uses a priority encoder to select the window to use for color key compare. The order of priority is Window A, then B, then C when multiple windows are enabled for color key.

Figure 89 Keying Example with Key Region Values

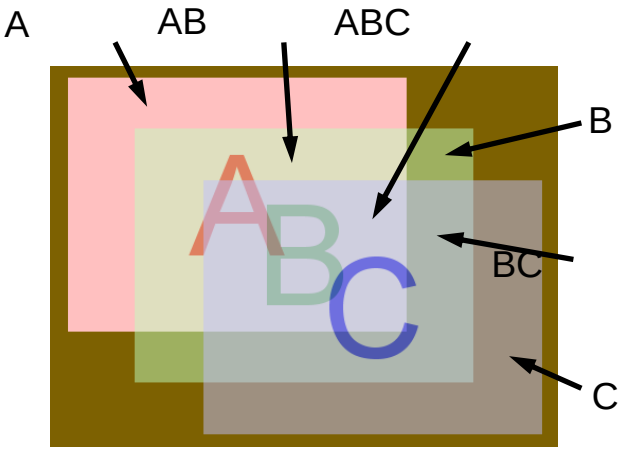
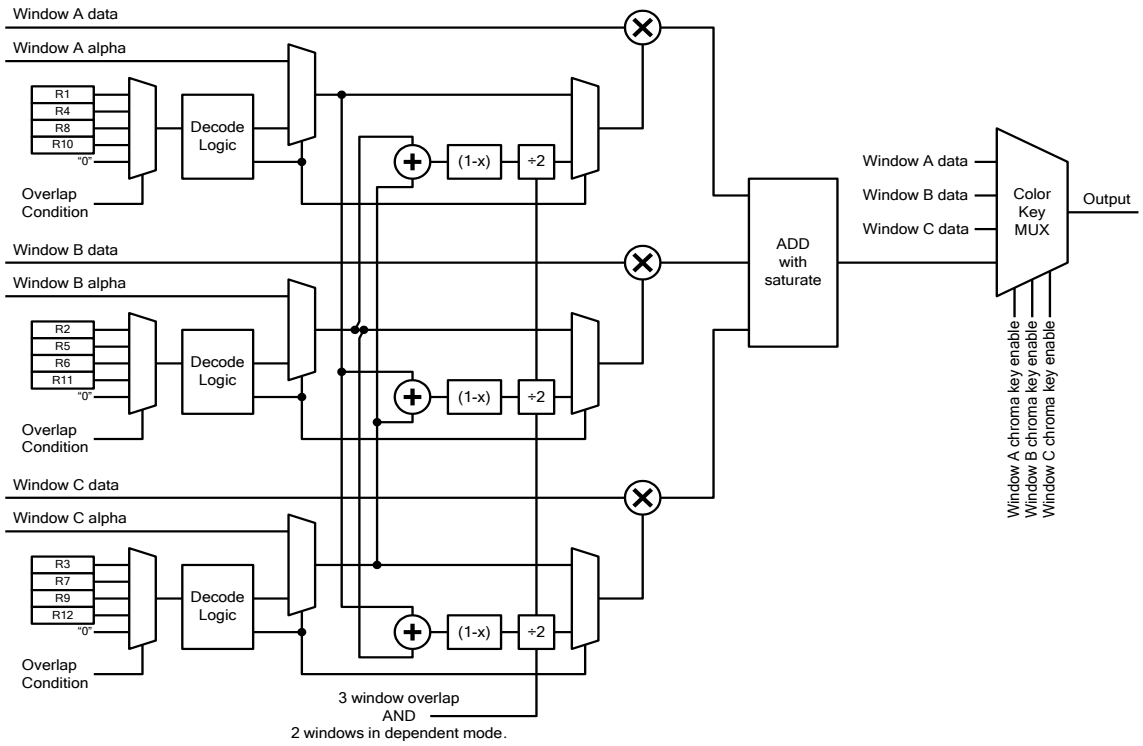


Table 99 Key Region Values for Keying Example

Region	A	B	C
A	100		
B		50	
C			50
AB	50	50	
AC	n/a	n/a	n/a
BC		25	50
ABC	25	25	50

Figure 90. Block Diagram of the Key Logic



29.2.3 Display Transformation

The Display is responsible for incrementing (and/or) decrementing *x*-direction scanning, and incrementing (and/or) decrementing *y*-direction scanning. By itself, the Display is not capable of doing the line scanning in the *y*-direction, which is needed for 90-degree or 270-degree display rotation.

29.2.4 Dual Display

There are two identical Display controllers, Display and Display B, instantiated in the chip, allowing the chip to drive two panels simultaneously. Because of pad limitation, Display B only supports a subset of the LCD interfaces that Display supports when used simultaneously. In dual display mode, Display and Display B may be programmed to non 1-clock/1-pixel modes such that the maximum data pins needed is 9-bit (refer to table).

For dual LCD display, there are two identical pad macros, one for each instance of display. By duplicating the pad macro, each display can drive the same type of interfaces as long as the pin is available. Register bits from within the host control whether the pin is driven by Display or Display B.

Each display controller may drive any of the available outputs, allowing such combinations as (but not limited to) LCD + TV or LCD + DSI.

Figure 91. Dual Display Block Diagram

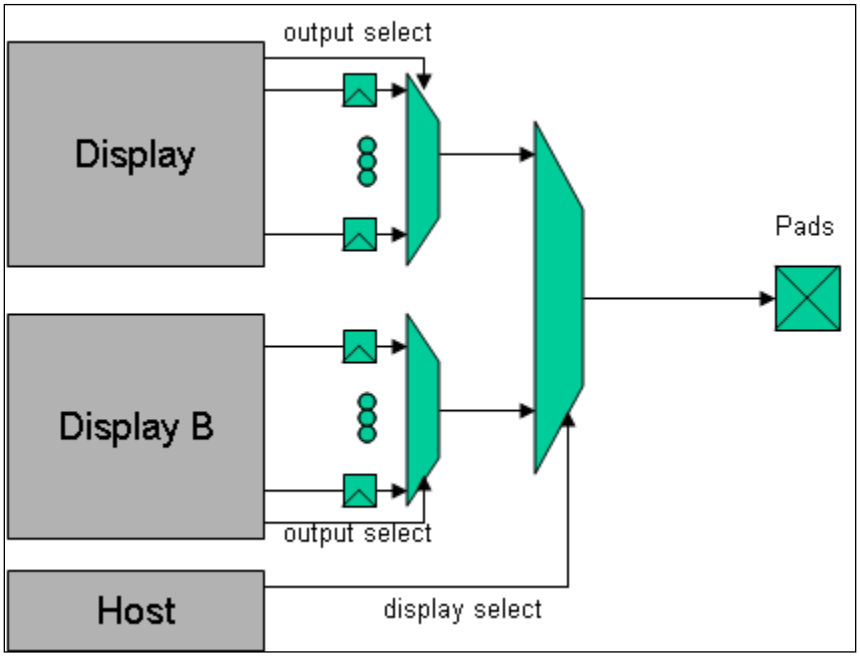


Table 100 shows which pairs of primary and secondary LCDs support independent simultaneous refresh of both LCDs.

Table 100: Display Pairs

Primary	Secondary	Simultaneous Dual Refresh?	Primary	Secondary	Simultaneous Dual Refresh?	Primary	Secondary	Simultaneous Dual Refresh?
24-bit RGB	9-bit CPU	No	16-bit RGB	9-bit CPU	Yes	9-bit CPU	9-bit CPU	Yes
	8-bit CPU	No		8-bit CPU	Yes		8-bit CPU	Yes
	5-wire SPI	Yes		5-wire SPI	Yes		5-wire SPI	Yes
24-bit RGB +5-wire SPI	9-bit CPU	No	16-bit RGB +5-wire SPI	9-bit CPU	Yes	8-bit CPU	9-bit CPU	Yes
	8-bit CPU	No		8-bit CPU	Yes		8-bit CPU	Yes
	5-wire SPI	Yes		5-wire SPI	Yes		5-wire SPI	Yes
18-bit RGB	9-bit CPU	Yes	18-bit CPU	9-bit CPU	Yes			
	8-bit CPU	Yes		8-bit CPU	Yes			
	5-wire SPI	Yes		5-wire SPI	Yes			
18-bit RGB +5-wire SPI	9-bit CPU	Yes	16-bit CPU	9-bit CPU	Yes			
	8-bit CPU	Yes		8-bit CPU	Yes			
	5-wire SPI	Yes		5-wire SPI	Yes			

29.3 VESA Timings

The VESA standard timings for Personal Computer Monitors can be roughly divided – for practical purposes – into two distinct groups: “Standard” and “Reduced Blanking”. Standard timing is used for CRT monitors and LCD panels. Reduced Blanking timings are used only for LCD panels that can accept Reduced Blanking timings.

The purpose of Reduced Blanking is to acknowledge the redundancy of the requirement of a large Horizontal Blanking period when outputting a video raster to an LCD panel. The Original purpose of the HB period was to allow a CRT time to move the electron beam from the right side of the screen to the left side of the screen prior to starting the next line. This takes a finite amount of time.

However in an LCD, this time can be made arbitrarily small with suitable care in the design of the panel. The advantage that Reduced Blanking gives is that the pixel clock frequency can be reduced for a given H. Total, V. Total and Vertical Frequency (refresh rate). This in turn results in the ability to either lower power consumption for a given resolution, or increase the resolution without exceeding the capabilities of the interface between the Display Controller and the Display Device. Both sets of timing parameters are provided in the following sections.

29.4 Television Timings

There is no formula for calculating the timing parameters of video standards used in television. Instead, each standard is governed by a specification document maintained by the appropriate standards body.

For professional broadcast television, the standards are maintained by the Society of Motion Picture and Television Engineers (SMPTE), the European Broadcast Union (EBU) and the International Telecommunication Union (ITU). For consumer electronic devices, many of the SMPTE standards have been adopted and modified to suit the slightly different needs of this market. These standards are governed by the Electronic Industries Alliance (EIA) and Consumer Electronics Association (CEA).

However – for the purposes of programming the raster structure in the Tegra® 2 Processor Series, the professional standards and the consumer standards can be thought of as being the same.

Some example timing parameters are shown in the following table. Note that these are just a rough guide – hence the reason the names are in quotation marks. Reference should be made to specific standards for precise details of what timing parameter values should be used.

Table 101. Television Standard Timing Parameters

	"NTSC"2	"PAL" 2	"480p"	"720p"	"1080i" 2	"1080p"
HA	720	720	720	1280	1920	1920
VA	487	576	483	720	1080	1080
FV	59.94	50	601	601	601	601
HFP	16	12	16	70	44	44
HS	63	63	63	804	884	884
HBP	59	69	59	220	148	148
VFP	33	2/33	6	5	2/33	4
VS	33	33	6	5	53	5
VBP	133	193	30	20	153	36
FP	13.50	13.50	27.001	74.251	74.251	148.501

Note: These standards also come in versions where the pixel clock frequency (FP) is divided by 1.001 to give a vertical frame rate (FV) that is also 1:1.001 slower. All other parameters are identical. Note that in the Tegra 2 Processor Series, no distinction is made between the two slightly different clock frequencies since the difference falls within the specified clock frequency tolerance for each variety.

NTSC, PAL and 1080i are all interlaced standards and as such are not supported by the raster generator in the pixel pipeline of Tegra 2 Processor Series, which only understands progressive rasters. For these standards, FV is given as the field rate, where there are two fields per frame. NTSC and PAL are supported by the dedicated TV encoder module in Tegra 2 Processor Series devices.

Since these standards are interlaced, some parameters are different for field 1 and field 2. Also – strictly speaking – some of these parameters should consist of half-lines.

HD formats from 720p and above have tri-level sync pulses. The HS figure represents the total width of the tri-level sync from first falling edge to second falling edge.

29.5 Programming

29.5.1 Raster Generator

The following is a list of register names that correspond to the abbreviations used throughout this document.

Table 102. Abbreviation / Register Field Mapping

Abbreviation	Method/Register	Field
HFP	FRONT_PORCH	H_FRONT_PORCH
HS	SYNC_WIDTH	H_SYNC_WIDTH
HBP	BACK_PORCH	H_BACK_PORCH
HA	DISP_ACTIVE	H_DISP_ACTIVE
VFP	FRONT_PORCH	V_FRONT_PORCH
VS	SYNC_WIDTH	V_SYNC_WIDTH
VBP	BACK_PORCH	V_BACK_PORCH
VA	DISP_ACTIVE	V_DISP_ACTIVE

HT and VT are derived from the other registers by summing together all the related periods:

HT = HFP + HS + HBP + HA

VT = VFP + VS + VBP + VA

29.6 Display Controller Register Definition

Display supports three windows: window A, window B, window C

Window A is a graphics only window (does not support YUV data format) and it supports only simple non-filtered up/down scaling via pixel and line replication/deletion.

Window B is an overlay window which supports both RGB and YUV data format and it supports filtered up/down scaling. Vertical filter is 2 tap with 16-phase resolution and horizontal filter is 6-tap with 16-phase resolution.

Window C is an overlay window which supports both RGB and YUV data format and it supports filtered up/down scaling. Horizontal filter is 6-tap with 16-phase resolution.

The Display Controller register consists of two template files:

- ardisplay_TEMPLATE.spec: these registers are applicable to all display windows
- ardisplay_b_TEMPLATE.spec: these registers are applicable to windows, A, B, or C. There are three copies of these registers in DISPLAY. Window A, B, and C registers are copies of each other, except for differences in window features. The copies of a register in window A, B, and C share the same address. Register field DISPLAY_WINDOW_HEADER is used to control whether the subsequent programming goes to window A, B, or C registers, or any combination of them. Before reading these registers, DISPLAY_WINDOW_HEADER must be programmed to enable only one window, so that DISPLAY knows which register copy is being read.

Color Palette

These appear as three instances of 256 32-bit registers with each register consisting of one RGB pixel so not all 32 bits in the registers are used. One instance is used for window A, another instance is used for window B, and the third instance is used for window C. In reality, each instance is implemented as triple 256-words dual-port register file (2P RF) with one read port and one write port (1R1W) and with each word consisting of 1 red/green/blue component. Read port of the color palettes are used for the corresponding window A, window B, or window C and write port of the color palettes are used for host write.

Note that the host cannot read these color palettes so it must cache this color palette somewhere else to be able to read them. This is done to reduce area.

29.6.1 Display Shadow Registers

Display registers have three shadow types:

1. Not Shadowed

   Writes to these registers take effect immediately.

2. Double Buffered

   Each register has two copies: ASSEMBLY and ACTIVE. ACTIVE is the working copy.

   These two copies share the same address/offset.

   WRITE_MUX can be programmed to choose which copy the subsequent register writes goes to:

   - When set to ACTIVE, both ACTIVE and ASSEMBLY copies will be written
   - When set to ASSEMBLY, only ASSEMBLY copy will be written

   READ_MUX can be programmed to choose which copy the subsequent register reads comes from.

   If display is in STOP mode, ASSEMBLY copy is latched into ACTIVE copy immediately after

   (GENERAL/WIN_A/WIN_B/WIN_C)_ACT_REQ is programmed. In other modes the latching happens on the next frame boundary after (GENERAL/WIN_A/WIN_B/WIN_C)_ACT_REQ is programmed.

3. Triple buffered

   Each register has three copies: ASSEMBLY, ARM, and ACTIVE. ACTIVE is the working copy.

   ASSEMBLY and ACTIVE copies share the same address/offset, the ARM copy is located at the address/offset one bigger than the ASSEMBLY/ACTIVE copy.

   WRITE_MUX can be programmed to choose which copy the subsequent register writes goes to:

   - When set to ACTIVE, all three copies will be written
   - When set to ASSEMBLY, only ASSEMBLY copy will be written

   READ_MUX can be programmed to choose which copy the subsequent register reads comes from.

   - To read back ASSEMBLY or ACTIVE copy, set READ_MUX correctly before register reads.

   These two copies share the same address.

   - To read back ARM copy, do not set READ_MUX, but read back the register/register-field with "_NS" in their names, the offset of which is 1 bigger than its ASSEMBLY/ACTIVE counterpart.

   ASSEMBLY copy is latched into ARM copy immediately after GENERAL/WIN_A/WIN_B/WIN_C)_UPDATE is programmed.

   If display is in STOP mode, ARM copy is latched into ACTIVE copy immediately after (GENERAL/WIN_A/WIN_B/WIN_C)_ACT_REQ is programmed. In other modes the latching happens on the next frame boundary after (GENERAL/WIN_A/WIN_B/WIN_C)_ACT_REQ is programmed.

29.6.2 Display Control Modes

The DISPLAY_COMMAND is used to set display control mode (**CONTINUOUS**, **ONE-SHOT**, or **STOP**).

Display switches mode when CTROL_MODE is programmed to shadow register and then activated.

Display enters a mode when the register activation happens, either on frame boundary when entering **STOP** mode, or immediately when exiting **STOP** mode.

In **STOP** mode, display is in idle. Pixel clock is not running.

In **CONTINUOUS** mode, display keeps refreshing the output frame by frame. This mode is mostly used for the panels without internal frame buffer.

In **ONE-SHOT** mode, display waits for a trigger before refreshing each frame. Between those frames, display is in idle. This mode is usually for the panels with internal frame buffer.

(TVO is an exception. Since TV does not have internal frame buffer, TVO works in continuous mode and controls the pace. DISPLAY is thus set to ONE-SHOT mode to work as slave of TVO. Other than that, it works much like CONTINUOUS mode.)

In ONE-SHOT mode, there are different types of triggers as follows:

- Input triggers

  Input trigger is to notify DISPLAY that a new memory surface is ready to be displayed.
- Host Trigger

  Host trigger is requested when NC_HOST_TRIG_ENABLE is programmed. This register field is in the same register as shadow registers UPDATE/ACT_REQ so that the trigger can happen atomically with register updates for the new frame. When host trigger is requested within a frame, the requested frame will start immediately after the end of the current frame.
- Peer Trigger

  Peer trigger is requested when a peer (VI/EPP/MPE/2D...) sends over a full set of buffer indexes for a complete frame. DISPLAY calculates the new buffer address based on the buffer index received. Whether DISPLAY sends a frame immediately depends on the value of WINDOW_A/B/C_NC_DISPLAY. If it is set to ENABLE, then DISPLAY sends a frame immediately (unless being held off by output triggers), otherwise DISPLAY does not send a new frame, though address change has already taken effect.
- Output Triggers

  Output trigger is to notify DISPLAY that a panel is ready to receive data from DISPLAY.
- TVO Triggers

  When TVO is enabled, DISPLAY waits until TVO sends over a vsync pulse, then sends a new frame, regardless of whether an input trigger has happened.
- Tear Effect Signal Triggers

  When MSF/SSF register field is set to ENABLE, DISPLAY holds off a frame requested by input triggers until DISPALY receives a "ready" from the pins that connect to the tearing effect signals from the panel, then sends a new frame to panel. However, if no input trigger has been requested, DISPLAY does not send a new frame. That is different from the behavior of TVO triggers.

In CONTINUOUS mode, the meaning of the triggers is different.

- Input Triggers

  Since in CONTINUOUS mode trigger happen automatically and repeatedly, trigger here only affects the buffer address change on a window.
- Host Trigger

Host can change the buffer address and then write WIN_A/B/C_ACT_REQ to request the buffer address change. Note that for syncpt logic, it cannot tell if a WIN_A/B/C register activation indicates the change of address, so it behaves always like a change of address happens on any register activation requested by WIN_A/B/C_ACT_REQ. As the result, RD_DONE/OP_DONE is returned.

- Peer Trigger

  When a peer (VI/EPP/MPE/2D...) sends over a full set of buffer indexes for a complete frame, the address is calculated and is used in next frame.

- Output triggers

  These triggers are not applicable in CONTINUOUS mode. TVO_ENABLE and MSF/SSF_ENABLE should never be set in CONTINUOUS mode.

29.6.3 Sync Points

DISPLAY has 4 syncpt clients:

- GENERAL

  Conditions

  - 0: IMMEDIATE return INDX immediately
  - 1: OP_DONE not meaningful, same as IMMEDIATE
  - 2: RD_DONE not meaningful, same as IMMEDIATE
  - 3: REG_WR_SAFE returns INDX whenever it is safe to program GENERAL shadow registers (when all previous GENERAL activation has happened)
  - 4: HSPI returns INDX whenever it is safe to send HSPI data (when all HSPI_*** registers are safe to be programmed)
  - 5: FRAME_DONE returns INDX on the next frame end
  - 6: VPULSE3 returns INDX on the next vpulse3 leading edge (for timing sensitive operations if needed)
  - 7: FRAME_START returns INDX on the next frame start (currently for hardware testing, but can be used if needed)

- WIN_A

  Conditions

  - 0: IMMEDIATE return INDX immediately
  - 1: OP_DONE same as RD_DONE
  - 2: RD_DONE returns INDX when all WIN_A reads for frames activated before the INCR_SYNCPT are complete. In ONE-SHOT mode with TVO disabled, this happens after the last row of window A display. In CONTINUOUS mode, or in ONE-SHOT mode with TVO enabled, this happens on the last row of window A or on frame end (depending if the INCR_SYNCPT is issued before or after the window A last row).
  - 3: REG_WR_SAFE returns INDX whenever it is safe to program WIN_A shadow registers (when all previous WIN_A activation has happened)

- WIN_B

  Similar to WIN_A

- WIN_C

  Similar to WIN_A

- DISPLAY Continuous syncpt

  VSYNC: When enabled, return syncpt whenever a vblank start happens.

29.6.4 Raise Actions (Legacy)

The Raise action is used to check the state of the display's command execution. The Raise action can be enabled by itself or with other actions.

The raise value is returned to the host controller if the current and all previous commands have completed execution. If the command is executed with the delayed mode, the raise is not returned till that command is executed at the end of the frame.

If there is no pending command when the Raise is issued, the raise value is returned immediately. The raise value is a 4-bit number included in the command.

Context switch acknowledge, register reads, and raises are returned to the host through a read FIFO. However, because there may be more than one raise, read, or acknowledge available for writing to the FIFO in a certain cycle, there must be a priority encoder. Here is the priority:

1. Context switch
2. Register read
3. SIGNAL_RAISE
4. SIGNAL_RAISE1
5. SIGNAL_RAISE2
6. SIGNAL_RAISE3
7. DISP_COMMAND_RAISE
8. HSPI_RAISE
9. Refcount

These packets are for verification; they have no relevance to SW.

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.

29.7 Display CMD Registers

29.7.1 DC_CMD_GENERAL_INCR_SYNCPT_0

Offset: 000h | Read/Write: R/W | Reset: 0b0000000000000000

Bit	Reset	Description
15:8	0x0	GENERAL_COND: Condition mapped from raise/wait 0 = IMMEDIATE 1 = OP_DONE 2 = RD_DONE 3 = REG_WR_SAFE 4 = HSPI 5 = FRAME_DONE 6 = VPULSE3 7 = FRAME_START 8 = COND_8 9 = COND_9 10 = COND_10 11 = COND_11 12 = COND_12 13 = COND_13 14 = COND_14 15 = COND_15
7:0	0x0	GENERAL_INDX: syncpt index value

29.7.2 DC_CMD_GENERAL_INCR_SYNCPT_CNTRL_0

Offset: 001h | Read/Write: R/W | Reset: 0b0xxxxxxx0

Bit	Reset	Description
8	0x0	GENERAL_INCR_SYNCPT_NO_STALL: If NO_STALL is 1, then when fifos are full, INCR_SYNCPT methods will be dropped and the INCR_SYNCPT_ERROR[COND] bit will be set. If NO_STALL is 0, then when fifos are full, the client host interface will be stalled.
0	0x0	GENERAL_INCR_SYNCPT_SOFT_RESET: If SOFT_RESET is set, then all internal state of the client syncpt block will be reset. To do soft reset, first set SOFT_RESET of all host1x clients affected, then clear all SOFT_RESETs.

29.7.3 DC_CMD_GENERAL_INCR_SYNCPT_ERROR_0

Offset: 002h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	_NONE_	GENERAL_COND_STATUS: COND_STATUS[COND] is set if the fifo for COND overflows This bit is sticky and will remain set until overwritten with a zero

Reserve locations for future expansion

29.7.4 DC_CMD_WIN_A_INCR_SYNCPT_0

Offset: 008h | Read/Write: R/W | Reset: 0b0000000000000000

Bit	Reset	Description
15:8	0x0	WIN_A_COND: Condition mapped from raise/wait 0 = IMMEDIATE 1 = OP_DONE 2 = RD_DONE 3 = REG_WR_SAFE 4 = COND_4 5 = COND_5 6 = COND_6 7 = COND_7 8 = COND_8 9 = COND_9 10 = COND_10 11 = COND_11 12 = COND_12 13 = COND_13 14 = COND_14 15 = COND_15
7:0	0x0	WIN_A_INDX: syncpt index value

29.7.5 DC_CMD_WIN_A_INCR_SYNCPT_CNTRL_0

Offset: 009h | Read/Write: R/W | Reset: 0b0xxxxxx0

Bit	Reset	Description
8	0x0	WIN_A_INCR_SYNCPT_NO_STALL: If NO_STALL is 1, then when fifos are full, INCR_SYNCPT methods will be dropped and the INCR_SYNCPT_ERROR[COND] bit will be set. If NO_STALL is 0, then when fifos are full, the client host interface will be stalled.
0	0x0	WIN_A_INCR_SYNCPT_SOFT_RESET: If SOFT_RESET is set, then all internal state of the client syncpt block will be reset. To do soft reset, first set SOFT_RESET of all host1x clients affected, then clear all SOFT_RESETs.

29.7.6 DC_CMD_WIN_A_INCR_SYNCPT_ERROR_0

Offset: 00ah | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	_NONE_	WIN_A_COND_STATUS: COND_STATUS[COND] is set if the fifo for COND overflows This bit is sticky and will remain set until overwritten with a zero

Reserve locations for future expansion

29.7.7 DC_CMD_WIN_B_INCR_SYNCPT_0

Offset: 010h | Read/Write: R/W | Reset: 0b0000000000000000

Bit	Reset	Description
15:8	0x0	WIN_B_COND: Condition mapped from raise/wait 0 = IMMEDIATE 1 = OP_DONE 2 = RD_DONE 3 = REG_WR_SAFE 4 = COND_4 5 = COND_5 6 = COND_6 7 = COND_7 8 = COND_8 9 = COND_9 10 = COND_10 11 = COND_11 12 = COND_12 13 = COND_13 14 = COND_14 15 = COND_15
7:0	0x0	WIN_B_INDX: syncpt index value

29.7.8 DC_CMD_WIN_B_INCR_SYNCPT_CNTRL_0

Offset: 011h | Read/Write: R/W | Reset: 0b0xxxxxx0

Bit	Reset	Description
8	0x0	WIN_B_INCR_SYNCPT_NO_STALL: If NO_STALL is 1, then when fifos are full, INCR_SYNCPT methods will be dropped and the INCR_SYNCPT_ERROR[COND] bit will be set. If NO_STALL is 0, then when fifos are full, the client host interface will be stalled.
0	0x0	WIN_B_INCR_SYNCPT_SOFT_RESET: If SOFT_RESET is set, then all internal state of the client syncpt block will be reset. To do soft reset, first set SOFT_RESET of all host1x clients affected, then clear all SOFT_RESETs.

29.7.9 DC_CMD_WIN_B_INCR_SYNCPT_ERROR_0

Offset: 012h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	_NONE_	WIN_B_COND_STATUS: COND_STATUS[COND] is set if the fifo for COND overflows This bit is sticky and will remain set until overwritten with a zero

Reserve locations for future expansion

29.7.10 DC_CMD_WIN_C_INCR_SYNCPT_0

Offset: 018h | Read/Write: R/W | Reset: 0b0000000000000000

Bit	Reset	Description
15:8	0x0	WIN_C_COND: Condition mapped from raise/wait 0 = IMMEDIATE 1 = OP_DONE 2 = RD_DONE 3 = REG_WR_SAFE 4 = COND_4 5 = COND_5 6 = COND_6 7 = COND_7 8 = COND_8 9 = COND_9 10 = COND_10 11 = COND_11 12 = COND_12 13 = COND_13 14 = COND_14 15 = COND_15
7:0	0x0	WIN_C_INDX: syncpt index value

29.7.11 DC_CMD_WIN_C_INCR_SYNCPT_CNTRL_0

Offset: 019h | Read/Write: R/W | Reset: 0b0xxxxxxx0

Bit	Reset	Description
8	0x0	WIN_C_INCR_SYNCPT_NO_STALL: If NO_STALL is 1, then when fifos are full, INCR_SYNCPT methods will be dropped and the INCR_SYNCPT_ERROR[COND] bit will be set. If NO_STALL is 0, then when fifos are full, the client host interface will be stalled.
0	0x0	WIN_C_INCR_SYNCPT_SOFT_RESET: If SOFT_RESET is set, then all internal state of the client syncpt block will be reset. To do soft reset, first set SOFT_RESET of all host1x clients affected, then clear all SOFT_RESETs.

29.7.12 DC_CMD_WIN_C_INCR_SYNCPT_ERROR_0

Offset: 01ah | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	_NONE_	WIN_C_COND_STATUS: COND_STATUS[COND] is set if the fifo for COND overflows This bit is sticky and will remain set until overwritten with a zero

Reserve locations for future expansion

29.7.13 DC_CMD_CONT_SYNCPT_VSYNC_0

Offset: 028h | Read/Write: R/W | Reset: 0b0xxxxxxxx

Bit	Reset	Description
8	0x0	VSYNC_EN: on host read bus every time VSYNC (V-blank leading edge) happens and VSYNC_EN is set 0 = DISABLE 1 = ENABLE
7:0	_NONE_	VSYNC_INDX: return INDX (set HOST_CLRD packet TYPE field to SYNCPT)

29.7.14 DC_CMD_CTXSW_0

Context switch registers for class and channel

Should be common to all modules. Includes the current channel/class (which is writable by SW) and the next channel/class (which the hardware sets when it receives a context switch).

Context 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: 030h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxx11111x0000000000

Bit	Reset	Description
31:28	0xf	NEXT_CHANNEL: Next requested channel
25:16	0x0	NEXT_CLASS: Next requested class
15:12	0xf	CURR_CHANNEL: Current working channel, reset to 'invalid'
11	0x1	AUTO_ACK: Automatically acknowledge any incoming context switch requests 0 = MANUAL 1 = AUTOACK
9:0	0x0	CURR_CLASS: Current working class

29.7.15 DC_CMD_DISPLAY_COMMAND_OPTION0_0

Display Controller Option 0. This register is not effective until DISPLAY_COMMAND is written.

Class: Display Command

Offset: 031h | Read/Write: R/W | Reset: 0b000xxxxxxxx00000000

Bit	Reset	Description
18	0x0	WINDOW_C_NC_DISPLAY: Window C Non-Continuous Display This is effective only in Non-Continuous Display mode when window B buffer switching is not controlled by host. If this bit is enabled, a frame is sent whenever Window B buffer is switched. 0 = DISABLE 1 = ENABLE
17	0x0	WINDOW_B_NC_DISPLAY: Window B Non-Continuous Display This is effective only in Non-Continuous Display mode when window B buffer switching is not controlled by host. If this bit is enabled, a frame is sent whenever Window B buffer is switched. 0 = DISABLE 1 = ENABLE
16	0x0	WINDOW_A_NC_DISPLAY: Window A Non-Continuous Display This is effective only in Non-Continuous Display mode when window A buffer switching is not controlled by host. If this bit is enabled, a frame is sent whenever Window A buffer is switched. 0 = DISABLE 1 = ENABLE
7:6	0x0	SSF_SOURCE: Source pin for the SSF input Controls which pin will be used as the source for the trigger input when MSF mode is enabled. Note that although the same pins are available for both MSF and SSF, the order in the enum and hence the values differ between the pins. This is to maintain backwards compatibility with previous chips, which had a fixed mapping. The init value and the first value in the enum reflects this historical mapping. 0= LCD_DC pin (legacy default) 1= LCD_SPI pin 2= LCD_SDI pin 3= RESERVED for future use. 0 = SSF_LDC 1 = SSF_LSPI 2 = SSF_LSDI
5	0x0	SSF_ENABLE: Sub-Display Stop Frame (SSF) input This is effective only in Non-Continuous Display mode 0= Disabled 1= Enabled When enabled, SSF signal can be input through LDC pin. When SSF is enabled a trigger to send a frame in Non-Continuous Display mode will be delayed until SSF is active. 0 = DISABLE 1 = ENABLE
4	0x0	SSF_POLARITY: Sub-Display Stop Frame (SSF) Polarity 0= Active high 1= Active low
3:2	0x0	MSF_SOURCE: Source pin for the MSF input Controls which pin will be used as the source for the trigger input when MSF mode is enabled. Note that although the same pins are available for both MSF and SSF, the order in the enum and hence the values differ between the pins. This is to maintain backwards compatibility with previous chips, which had a fixed mapping. The init value and the first value in the enum reflects this historical mapping. 0= LCD_SPI pin (legacy default) 1= LCD_DC pin 2= LCD_SDI pin 3= RESERVED for future use. 0 = MSF_LSPI 1 = MSF_LDC 2 = MSF_LSDI
1	0x0	MSF_ENABLE: Main-Display Stop Frame (MSF) input This is effective only in Non-Continuous Display mode 0= Disabled 1= Enabled When enabled, MSF signal can be input through LSPI pin. When MSF is enabled a trigger to send a frame in Non-Continuous Display mode will be delayed until MSF is active. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
0	0x0	MSF_POLARITY: Main-Display Stop Frame (MSF) Polarity 0= Active high 1= Active low

29.7.16 DC_CMD_DISPLAY_COMMAND_0

Display Command

Offset: 032h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxx00xxxx0

Bit	Reset	Description
30:27	X	DISP_COMMAND_RAISE_CHANNEL_ID: Display Command Channel ID
26:22	X	DISP_COMMAND_RAISE_VECTOR: Display Command Raise Vector This raise vector is at the next line or frame boundary, depending on GENERAL_ACT_CNTR_SEL
6:5	0x0	DISPLAY_CTRL_MODE: Display Controller Mode 0= Stop Display, this can be used to stop sending frame at the next frame boundary. This is automatically generated in Non-Continuous Display after sending one frame. If this is issued when display controller is already stopped then there is no frame sent. Raise vector (if raise is enabled) is also returned immediately. This command can also be used in non-continuous display mode to stop accepting non-host trigger conditions from other clients. 1= Continuous Display, the display controller will continuously send frame. Continuous display mode can be stopped by switching to Non-Continuous Display or by issuing Stop Display. 2= Non-Continuous Display, the display controller is forced to send one frame of each active display and then wait for the next time this command is issued or for other (non-host) trigger conditions to send frame. The sending of frames may be delayed by MSF or SSF input signals from the display device. If a Stop Display is issued while in non-continuous display mode then non-host trigger conditions will no longer be accepted until the next time Non-Continuous Display is issued 0 = STOP 1 = C_DISPLAY 2 = NC_DISPLAY
0	0x0	DISP_COMMAND_RAISE: Display Command Raise. Raise vector will be returned at the end of command completion 0 = DISABLE 1 = ENABLE

29.7.17 DC_CMD_SIGNAL_RAISE_0

When written, next occurrence of the selected SIGNAL will cause a RAISE to be sent to the host. Software must not write this register if a previous request (from previous write) is still outstanding. Added SIGNAL_RAISE_TYPE option so that multiple raises can be returned without software intervention. SIGNAL_RAISE1, SIGNAL_RAISE2, or SIGNAL_RAISE3 can be used if more than one source signal is needed.

Offset: 033h | Read/Write: R/W | Reset: 0bxxxxxxx0xxxxxxxxxxx

Bit	Reset	Description
19:16	X	SIGNAL_RAISE_CHANNEL_ID: Signal Raise Channel ID
12	0x0	SIGNAL_RAISE_TYPE: 0= Oneshot, single raise returned 1= Continuous, raise is returned persistently whenever raise event is true until this register is reprogrammed such that SIGNAL_RAISE_SELECT=NONE or SIGNAL_RAISE_TYPE=ONESHOT 0 = ONESHOT 1 = CONT

Bit	Reset	Description
10:8	X	SIGNAL_RAISE_SELECT: which signal to raise on 0= none, no raise sent back 1= Frame End signal 2= V Blank signal 3= V Pulse 3 signal 4= Rising edge of V Blank signal 5= Falling edge of V Blank signal 6= Rising edge of V Pulse 3 signal 7= Falling edge of V Pulse 3 signal 0 = NONE 1 = FRAME_END 2 = VBLANK 3 = VPULSE3 4 = VBLANK_START 5 = VBLANK_END 6 = VPULSE3_START 7 = VPULSE3_END
4:0	X	SIGNAL_RAISE_VECTOR: bit number to raise

29.7.18 DC_CMD_DISPLAY_POWER_CONTROL_0

Display Power Control

Offset: 036h | Read/Write: R/W | Reset: 0b00xxxxx0x0xxxxxxx0x0x0x0x0

Bit	Reset	Description
25	0x0	HSPI_ENABLE: Host SPI write cycle Enable. SPI_ENABLE must be enabled also for this bit to be effective. 0 = DISABLE 1 = ENABLE
24	0x0	SPI_ENABLE: SPI interface Enable. This enables clock to SPI interface logic for Host SPI, IS SPI, and LCD SPI. 0 = DISABLE 1 = ENABLE
18	0x0	PM1_ENABLE: PM1 signal Enable 0 = DISABLE 1 = ENABLE
16	0x0	PM0_ENABLE: PM0 signal Enable 0 = DISABLE 1 = ENABLE
8	0x0	PW4_ENABLE: PW4 signal Enable. This signal can be output at the pad for display power sequencing. 0 = DISABLE 1 = ENABLE
6	0x0	PW3_ENABLE: PW3 signal Enable. This signal can be output at the pad for display power sequencing. 0 = DISABLE 1 = ENABLE
4	0x0	PW2_ENABLE: PW2 signal Enable. This signal controls pixel data processing. It should be enabled during V blank time. This signal also controls the time when pin polarity takes effect at the pad. This signal can be output at the pad for display power sequencing. 0 = DISABLE 1 = ENABLE
2	0x0	PW1_ENABLE: PW1 signal Enable. This signal can be output at the pad for display power sequencing. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
0	0x0	PW0_ENABLE: PW0 signal Enable. This signal controls the display H and V counters. It must be enabled first and disabled last during display power sequencing. This signal can be output at the pad for display power sequencing. 0 = DISABLE 1 = ENABLE

29.7.19 DC_CMD_INT_STATUS_0

Interrupt Status. This reflects status of all pending interrupts which is valid as long as the interrupt is not cleared even if the interrupt is masked. A pending interrupt can be cleared by writing a '1' to this the corresponding interrupt status bit in this register.

Display Interrupt and Status

Offset: 037h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
20	X	GPIO_2_INT: GPIO 2 Interrupt Status, connected to LCD_PWR2 0= interrupt not pending 1= interrupt pending
19	X	GPIO_1_INT: GPIO 1 Interrupt Status, connected to LCD_PWR1 0= interrupt not pending 1= interrupt pending
18	X	GPIO_0_INT: GPIO 0 Interrupt Status, connected to LCD_PWR0 0= interrupt not pending 1= interrupt pending
16	X	WIN_C_OF_INT: Window C Overflow Interrupt Status 0= interrupt not pending 1= interrupt pending
15	X	WIN_B_OF_INT: Window B Overflow Interrupt Status 0= interrupt not pending 1= interrupt pending
14	X	WIN_A_OF_INT: Window A Overflow Interrupt Status 0= interrupt not pending 1= interrupt pending
13	X	SSF_INT: Sub-Display Stop Frame Interrupt Status 0= interrupt not pending 1= interrupt pending
12	X	MSF_INT: Main-Display Stop Frame Interrupt Status 0= interrupt not pending 1= interrupt pending
11	X	EPP_OF_INT: Display2epp Overflow Interrupt Status 0= interrupt not pending 1= interrupt pending
10	X	WIN_C_UF_INT: Window C Underflow Interrupt Status 0= interrupt not pending 1= interrupt pending

Bit	Reset	Description
9	X	WIN_B_UF_INT: Window B Underflow Interrupt Status 0= interrupt not pending 1= interrupt pending
8	X	WIN_A_UF_INT: Window A Underflow Interrupt Status 0= interrupt not pending 1= interrupt pending
7	X	SPI_BUSY_INT: SPI Busy Interrupt Status 0= interrupt not pending 1= interrupt pending
4	X	V_PULSE3_INT: Vertical Pulse 3 Interrupt 0= interrupt not pending 1= interrupt pending
3	X	H_BLANK_INT: Horizontal Blank Interrupt 0= interrupt not pending 1= interrupt pending
2	X	V_BLANK_INT: Vertical Blank Interrupt 0= interrupt not pending 1= interrupt pending
1	X	FRAME_END_INT: Frame End Interrupt 0= interrupt not pending 1= interrupt pending
0	X	CTXSW_INT: Context Switch Interrupt Status (this is cleared on write) 0= interrupt not pending 1= interrupt pending

29.7.20 DC_CMD_INT_MASK_0

Interrupt Mask Setting bits in this register masked the corresponding interrupt but does not clear a pending interrupt and does not prevent a pending interrupt to be generated. Masking an interrupt also does not clear a pending interrupt status and does not prevent a pending interrupt status to be generated.

Offset: 038h | Read/Write: R/W | Reset: 0b000x00000x0000xx00000

Bit	Reset	Description
20	0x0	GPIO_2_INT_MASK: GPIO 2 Interrupt Mask, connected to LCD_PWR2 0 = MASKED 1 = NOTMASKED
19	0x0	GPIO_1_INT_MASK: GPIO 1 Interrupt Mask, connected to LCD_PWR1 0 = MASKED 1 = NOTMASKED
18	0x0	GPIO_0_INT_MASK: GPIO 0 Interrupt Mask, connected to LCD_PWR0 0= interrupt masked 1= interrupt not masked 0 = MASKED 1 = NOTMASKED
16	0x0	WIN_C_OF_INT_MASK: Window C Overflow Interrupt Mask 0 = MASKED 1 = NOTMASKED

Bit	Reset	Description
15	0x0	WIN_B_OF_INT_MASK: Window B Overflow Interrupt Mask 0 = MASKED 1 = NOTMASKED
14	0x0	WIN_A_OF_INT_MASK: Window A Overflow Interrupt Mask 0 = MASKED 1 = NOTMASKED
13	0x0	SSF_INT_MASK: Sub-Display Stop Frame Interrupt Mask 0 = MASKED 1 = NOTMASKED
12	0x0	MSF_INT_MASK: Main-Display Stop Frame Interrupt Mask 0 = MASKED 1 = NOTMASKED
10	0x0	WIN_C_UF_INT_MASK: Window C Underflow Interrupt Mask 0 = MASKED 1 = NOTMASKED
9	0x0	WIN_B_UF_INT_MASK: Window B Underflow Interrupt Mask 0 = MASKED 1 = NOTMASKED
8	0x0	WIN_A_UF_INT_MASK: Window A Underflow Interrupt Mask 0 = MASKED 1 = NOTMASKED
7	0x0	SPI_BUSY_INT_MASK: SPI Busy Interrupt Mask 0 = MASKED 1 = NOTMASKED
4	0x0	V_PULSE3_INT_MASK: Vertical Pulse 3 Interrupt Mask 0 = MASKED 1 = NOTMASKED
3	0x0	H_BLANK_INT_MASK: Horizontal Blank Interrupt Mask 0 = MASKED 1 = NOTMASKED
2	0x0	V_BLANK_INT_MASK: Vertical Blank Interrupt Mask 0 = MASKED 1 = NOTMASKED
1	0x0	FRAME_END_INT_MASK: Frame End Interrupt Mask 0 = MASKED 1 = NOTMASKED
0	0x0	CTXSW_INT_MASK: Context Switch Interrupt Mask 0 = MASKED 1 = NOTMASKED

29.7.21 DC_CMD_INT_ENABLE_0

Interrupt Enable 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.

Offset: 039h | Read/Write: R/W | Reset: 0b000x00000x0000xx00001

Bit	Reset	Description
20	0x0	GPIO_2_INT_ENABLE: Display GPIO_2 Interrupt Enable, connected to LCD_PWR2 0 = DISABLE 1 = ENABLE
19	0x0	GPIO_1_INT_ENABLE: Display GPIO_1 Interrupt Enable, connected to LCD_PWR1 0 = DISABLE 1 = ENABLE
18	0x0	GPIO_0_INT_ENABLE: Display GPIO_0 Interrupt Enable, connected to LCD_PWR0 0 = DISABLE 1 = ENABLE
16	0x0	WIN_C_OF_INT_ENABLE: Window C Overflow Interrupt Enable 0 = DISABLE 1 = ENABLE
15	0x0	WIN_B_OF_INT_ENABLE: Window B Overflow Interrupt Enable 0 = DISABLE 1 = ENABLE
14	0x0	WIN_A_OF_INT_ENABLE: Window A Overflow Interrupt Enable 0 = DISABLE 1 = ENABLE
13	0x0	SSF_INT_ENABLE: Sub-Display Stop Frame Interrupt Enable 0 = DISABLE 1 = ENABLE
12	0x0	MSF_INT_ENABLE: Main-Display Stop Frame Interrupt Enable 0 = DISABLE 1 = ENABLE
10	0x0	WIN_C_UF_INT_ENABLE: Window C Underflow Interrupt Enable 0 = DISABLE 1 = ENABLE
9	0x0	WIN_B_UF_INT_ENABLE: Window B Underflow Interrupt Enable 0 = DISABLE 1 = ENABLE
8	0x0	WIN_A_UF_INT_ENABLE: Window A Underflow Interrupt Enable 0 = DISABLE 1 = ENABLE
7	0x0	SPI_BUSY_INT_ENABLE: SPI Busy Interrupt Enable 0 = DISABLE 1 = ENABLE
4	0x0	V_PULSE3_INT_ENABLE: Vertical Pulse 3 Interrupt Enable 0= interrupt masked 1= interrupt not masked 0 = DISABLE 1 = ENABLE
3	0x0	H_BLANK_INT_ENABLE: Horizontal Blank Interrupt Enable 0= interrupt masked 1= interrupt not masked 0 = DISABLE 1 = ENABLE
2	0x0	V_BLANK_INT_ENABLE: Vertical Blank Interrupt Enable 0= interrupt masked 1= interrupt not masked 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1	0x0	FRAME_END_INT_ENABLE: Frame End Interrupt Enable 0= interrupt masked 1= interrupt not masked 0 = DISABLE 1 = ENABLE
0	0x1	CTXSW_INT_ENABLE: Context Switch Interrupt Enable 0= interrupt disabled 1= interrupt enabled 0 = DISABLE 1 = ENABLE

29.7.22 DC_CMD_INT_TYPE_0

Interrupt Type Two interrupt types are available: a. Edge interrupt - transition on input signal/event generates pending interrupt b. Level interrupt - active level on input signal/event generates pending interrupt

Offset: 03ah | Read/Write: R/W | Reset: 0b000x00000x0000xx0000

Bit	Reset	Description
20	0x0	GPIO_2_INT_TYPE: Display GPIO_2 Interrupt Type, connected to LCD_PWR2 0 = EDGE 1 = LEVEL
19	0x0	GPIO_1_INT_TYPE: Display GPIO_1 Interrupt Type, connected to LCD_PWR1 0 = EDGE 1 = LEVEL
18	0x0	GPIO_0_INT_TYPE: Display GPIO_0 Interrupt Type, connected to LCD_PWR0 0 = EDGE 1 = LEVEL
16	0x0	WIN_C_OF_INT_TYPE: Window C Overflow Interrupt Type 0 = EDGE 1 = LEVEL
15	0x0	WIN_B_OF_INT_TYPE: Window B Overflow Interrupt Type 0 = EDGE 1 = LEVEL
14	0x0	WIN_A_OF_INT_TYPE: Window A Overflow Interrupt Type 0 = EDGE 1 = LEVEL
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

Bit	Reset	Description
7	0x0	SPI_BUSY_INT_TYPE: SPI Busy Interrupt Type 0 = EDGE 1 = LEVEL
4	0x0	V_PULSE3_INT_TYPE: Vertical Pulse 3 Interrupt Type 0 = EDGE 1 = LEVEL
3	0x0	H_BLANK_INT_TYPE: Horizontal Blank Interrupt Type 0 = EDGE 1 = LEVEL
2	0x0	V_BLANK_INT_TYPE: Vertical Blank Interrupt Type 0 = EDGE 1 = LEVEL
1	0x0	FRAME_END_INT_TYPE: Frame End Interrupt Type 0 = EDGE 1 = LEVEL

29.7.23 DC_CMD_INT_POLARITY_0

Interrupt Polarity For edge interrupt, these bits specify whether a pending interrupt is generated on falling edge or on rising edge of the corresponding input signal/event. For level interrupt, these bits specify whether a pending interrupt is generated on low level or on high level of the corresponding input signal/event. 0 rw CTXSW_INT_POLARITY init=0 // Context Switch Interrupt Polarity enum (LOW, HIGH) // 0= falling edge or low level interrupt // 1= rising edge or high level interrupt.

Offset: 03bh | Read/Write: R/W | Reset: 0b000x00000x0000xx0000

Bit	Reset	Description
20	0x0	GPIO_2_INT_POLARITY: Display GPIO_2 Interrupt. Interrupt Polarity, connected to LCD_PWR2 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
19	0x0	GPIO_1_INT_POLARITY: Display GPIO_1 Interrupt. Interrupt Polarity, connected to LCD_PWR1 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
18	0x0	GPIO_0_INT_POLARITY: Display GPIO_0 Interrupt. Interrupt Polarity, connected to LCD_PWR0 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
16	0x0	WIN_C_OF_INT_POLARITY: Window C Overflow. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
15	0x0	WIN_B_OF_INT_POLARITY: Window B Overflow. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
14	0x0	WIN_A_OF_INT_POLARITY: Window A Overflow. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH

Bit	Reset	Description
13	0x0	SSF_INT_POLARITY: Sub-Display Stop Frame. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
12	0x0	MSF_INT_POLARITY: Main-Display Stop Frame. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
10	0x0	WIN_C_UF_INT_POLARITY: Window C Underflow. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
9	0x0	WIN_B_UF_INT_POLARITY: Window B Underflow. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
8	0x0	WIN_A_UF_INT_POLARITY: Window A Underflow. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
7	0x0	SPI_BUSY_INT_POLARITY: SPI Busy. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
4	0x0	V_PULSE3_INT_POLARITY: V Pulse 3. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
3	0x0	H_BLANK_INT_POLARITY: H Blank. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
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

29.7.24 DC_CMD_SIGNAL_RAISE1_0

When written, next occurrence of the selected SIGNAL will cause a RAISE to be sent to the host. Software must not write this register if a previous request (from previous write) is still outstanding. Added SIGNAL_RAISE1_TYPE option so that multiple raises can be returned without software intervention.

Offset: 03ch | Read/Write: R/W | Reset: 0bxxxxxxx0xxxxxxxxxxx

Bit	Reset	Description
19:16	X	SIGNAL_RAISE1_CHANNEL_ID: Signal Raise Channel ID

Bit	Reset	Description
12	0x0	SIGNAL_RAISE1_TYPE: 0= Oneshot, single raise returned 1= Continuous, raise is returned persistently whenever raise event is true until this register is reprogrammed such that SIGNAL_RAISE1_SELECT=NONE or SIGNAL_RAISE1_TYPE=ONESHOT 0 = ONESHOT 1 = CONT
10:8	X	SIGNAL_RAISE1_SELECT: which signal to raise on 0= none, no raise sent back 1= Frame End signal 2= V Blank signal 3= V Pulse 3 signal 4= Rising edge of V Blank signal 5= Falling edge of V Blank signal 6= Rising edge of V Pulse 3 signal 7= Falling edge of V Pulse 3 signal 0 = NONE 1 = FRAME_END 2 = VBLANK 3 = VPULSE3 4 = VBLANK_START 5 = VBLANK_END 6 = VPULSE3_START 7 = VPULSE3_END
4:0	X	SIGNAL_RAISE1_VECTOR: bit number to raise

29.7.25 DC_CMD_SIGNAL_RAISE2_0

When written, next occurrence of the selected SIGNAL will cause a RAISE to be sent to the host. Software must not write this register if a previous request (from previous write) is still outstanding. Added SIGNAL_RAISE2_TYPE option so that multiple raises can be returned without software intervention.

Offset: 03dh | Read/Write: R/W | Reset: 0bxxxxxxx0xxxxxxxxxxx

Bit	Reset	Description
19:16	X	SIGNAL_RAISE2_CHANNEL_ID: Signal Raise Channel ID
12	0x0	SIGNAL_RAISE2_TYPE: 0= Oneshot, single raise returned 1= Continuous, raise is returned persistently whenever raise event is true until this register is reprogrammed such that SIGNAL_RAISE2_SELECT=NONE or SIGNAL_RAISE2_TYPE=ONESHOT 0 = ONESHOT 1 = CONT
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

29.7.26 DC_CMD_SIGNAL_RAISE3_0

When written, next occurrence of the selected SIGNAL will cause a RAISE to be sent to the host.

Software must not write this register if a previous request (from previous write) is still outstanding.

Added SIGNAL_RAISE3_TYPE option so that multiple raises can be returned without software intervention.

Offset: 03eh | Read/Write: R/W | Reset: 0bxxxxxxx0xxxxxxxxxxx

Bit	Reset	Description
19:16	X	SIGNAL_RAISE3_CHANNEL_ID: Signal Raise Channel ID
12	0x0	SIGNAL_RAISE3_TYPE: 0= Oneshot, single raise returned 1= Continuous, raise is returned persistently whenever raise event is true until this register is reprogrammed such that SIGNAL_RAISE3_SELECT=NONE or SIGNAL_RAISE3_TYPE=ONESHOT 0 = ONESHOT 1 = CONT
10:8	X	SIGNAL_RAISE3_SELECT: which signal to raise on 0= none, no raise sent back 1= Frame End signal 2= V Blank signal 3= V Pulse 3 signal 4= Rising edge of V Blank signal 5= Falling edge of V Blank signal 6= Rising edge of V Pulse 3 signal 7= Falling edge of V Pulse 3 signal 0 = NONE 1 = FRAME_END 2 = VBLANK 3 = VPULSE3 4 = VBLANK_START 5 = VBLANK_END 6 = VPULSE3_START 7 = VPULSE3_END
4:0	X	SIGNAL_RAISE3_VECTOR: bit number to raise

29.7.27 DC_CMD_STATE_ACCESS_0

Double/triple buffers read and write access control

Offset: 040h | Read/Write: R/W | Reset: 0b0x0

Bit	Reset	Description
2	0x0	WRITE_MUX: Write access control 0= write assembly state 1= write active state When set to ACTIVE, register writes also propagate to assembly set for double buffered registers, to both assembly and arm set for triple buffered registers. 0 = ASSEMBLY 1 = ACTIVE
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

29.7.28 DC_CMD_STATE_CONTROL_0

State Control for activating/arming new register state

Restrictions: ACT_REQ cannot be programmed at the same time the corresponding "UPDATE" is programmed.

If so desired, it should be split into two consecutive writes to this register.

Offset: 041h | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxx0000xxxx0000

Bit	Reset	Description
24	0x0	NC_HOST_TRIG_ENABLE: Host trigger enable. Effective only in Non-continuous mode. The exception is that when TVO is enabled, this trigger is ignored so as not to corrupt TV output. Note that when this field is enabled, GENERAL_ACT_REQ must be enabled at the same time. 0= disable: no frame is triggered 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
11	0x0	WIN_C_UPDATE: Trigger for arming state (from assembly to armed state) for the win C subset of the triple buffered registers. This register is also known as "Update Register" 0 = DISABLE 1 = ENABLE
10	0x0	WIN_B_UPDATE: Trigger for arming state (from assembly to armed state) for the win B subset of the triple buffered registers. This register is also known as "Update Register" 0 = DISABLE 1 = ENABLE
9	0x0	WIN_A_UPDATE: Trigger for arming state (from assembly to armed state) for the win A subset of the triple buffered registers. This register is also known as "Update Register" 0 = DISABLE 1 = ENABLE
8	0x0	GENERAL_UPDATE: Trigger for arming state (from assembly to armed state) for a subset of the triple buffered registers. This register is also known as "Update Register" 0 = DISABLE 1 = ENABLE
3	0x0	WIN_C_ACT_REQ: Window C activation request 0= no req pending/req completed 1= activation requested/pending 0 = DISABLE 1 = ENABLE
2	0x0	WIN_B_ACT_REQ: Window B activation request 0= no req pending/req completed 1= activation requested/pending 0 = DISABLE 1 = ENABLE
1	0x0	WIN_A_ACT_REQ: Window A activation request 0= no req pending/req completed 1= activation requested/pending 0 = DISABLE 1 = ENABLE
0	0x0	GENERAL_ACT_REQ: Non-window-specific 0= no req pending/req completed 1= activation requested/pending 0 = DISABLE 1 = ENABLE

29.7.29 DC_CMD_DISPLAY_WINDOW_HEADER_0

Display Window Header for programming display windows and their corresponding buffer start addresses.

Class: Display Window Programming Header

Offset: 042h | Read/Write: R/W | Reset: 0b000

Bit	Reset	Description
6	0x0	WINDOW_C_SELECT: Window C Select 0= disable window C programming 1= enable window C programming 0 = DISABLE 1 = ENABLE
5	0x0	WINDOW_B_SELECT: Window B Select 0= disable window B programming 1= enable window B programming 0 = DISABLE 1 = ENABLE

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

29.7.30 DC_CMD_REG_ACT_CONTROL_0

Register activation options

Offset: 043h | Read/Write: R/W | Reset: 0b0x0x0x0

Bit	Reset	Description
6	0x0	WIN_C_ACT_CNTR_SEL: Select which counter to use for window C activation 0 = VCOUNTER 1 = HCOUNTER
4	0x0	WIN_B_ACT_CNTR_SEL: Select which counter to use for window B activation 0 = VCOUNTER 1 = HCOUNTER
2	0x0	WIN_A_ACT_CNTR_SEL: Select which counter to use for window A activation 0 = VCOUNTER 1 = HCOUNTER
0	0x0	GENERAL_ACT_CNTR_SEL: Select which counter to use for general activation 0 = VCOUNTER 1 = HCOUNTER

29.8 Display COM Registers

29.8.1 DC_COM_CRC_CONTROL_0

CRC Control

CRC is provided for at speed testing and diagnostic. When CRC is enabled, the CRC logic waits for the next VSync pulse or the one after that (depending on CRC_WAIT) and then it captures one frame of data at the end of display pipeline and computes the CRC value.

After one frame of data is captured, the CRC logic will stop capturing data.

When CRC_INTPU_DATA = FULLL_FRAME, DISPLAY_COMMAND.DISPLAY_CTRL_MODE should be programmed to C_DISPLAY so that CRC works properly.

When CRC_INTPU_DATA = ACTIVE_DATA, it can work on both NC_DISPLAY and C_DISPLAY modes, and can work for multiple frames if CRC is checked, disabled, and re-enabled after the end of frame v-active area and before next vsync.

CRC logic takes 8-bit of control signals and 24-bit RGB pixel after dither and after display color (R and B) swap option.

Input [31:0] into CRC depends on CRC_INPUT_DATA. If programmed as FULL_FRAME, input data is {LCD_D20, LPV0, LCD_D19, LCD_D18, LCD_D21, VSYNC, HSYNC, ACTIVE, R[7:0], G[7:0], B[7:0]} and CRC runs over the entire frame (including blank). If programmed as ACTIVE_DATA, input data is {R[7:0], G[7:0], B[7:0]} and CRC runs only during active display area.

Offset: 300h | Read/Write: R/W | Reset: 0b0000

Bit	Reset	Description

Bit	Reset	Description
3	0x0	CRC_ALWAYS: CRC always: calculate CRC for every following frames 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 Vsync
0	0x0	CRC_ENABLE: CRC Enable 0 = DISABLE 1 = ENABLE

29.8.2 DC_COM_CRC_CHECKSUM_0

CRC Checksum

This register can be read by host after CRC logic stops capturing data.

Offset: 301h | Read/Write: RO | Reset: 0x00000000

Bit	Reset	Description
31:0	0x0	CRC_CHECKSUM: CRC Checksum

29.8.3 DC_COM_PIN_OUTPUT_ENABLE0_0

Pin Output Enable 0

Pin Output Enable registers

Offset: 302h | Read/Write: R/W | Reset: 0b0x0x0x0x0x0x0x0x0x0x0x0x0x0x0x0x

Bit	Reset	Description
30	0x0	LD15_OUTPUT_ENABLE: LD15 pin output enable 0 = ENABLE 1 = DISABLE
28	0x0	LD14_OUTPUT_ENABLE: LD14 pin output enable 0 = ENABLE 1 = DISABLE
26	0x0	LD13_OUTPUT_ENABLE: LD13 pin output enable 0 = ENABLE 1 = DISABLE
24	0x0	LD12_OUTPUT_ENABLE: LD12 pin output enable 0 = ENABLE 1 = DISABLE
22	0x0	LD11_OUTPUT_ENABLE: LD11 pin output enable 0 = ENABLE 1 = DISABLE

Bit	Reset	Description
20	0x0	LD10_OUTPUT_ENABLE: LD10 pin output enable 0 = ENABLE 1 = DISABLE
18	0x0	LD9_OUTPUT_ENABLE: LD9 pin output enable 0 = ENABLE 1 = DISABLE
16	0x0	LD8_OUTPUT_ENABLE: LD8 pin output enable 0 = ENABLE 1 = DISABLE
14	0x0	LD7_OUTPUT_ENABLE: LD7 pin output enable 0 = ENABLE 1 = DISABLE
12	0x0	LD6_OUTPUT_ENABLE: LD6 pin output enable 0 = ENABLE 1 = DISABLE
10	0x0	LD5_OUTPUT_ENABLE: LD5 pin output enable 0 = ENABLE 1 = DISABLE
8	0x0	LD4_OUTPUT_ENABLE: LD4 pin output enable 0 = ENABLE 1 = DISABLE
6	0x0	LD3_OUTPUT_ENABLE: LD3 pin output enable 0 = ENABLE 1 = DISABLE
4	0x0	LD2_OUTPUT_ENABLE: LD2 pin output enable 0 = ENABLE 1 = DISABLE
2	0x0	LD1_OUTPUT_ENABLE: LD1 pin output enable 0 = ENABLE 1 = DISABLE
0	0x0	LD0_OUTPUT_ENABLE: LD0 pin output enable 0 = ENABLE 1 = DISABLE

29.8.4 DC_COM_PIN_OUTPUT_ENABLE1_0

Pin Output Enable 1

Offset: 303h | Read/Write: R/W | Reset: 0b0x0x0x0xxx0x0x0xxxxxxxxxxxxx0x0

Bit	Reset	Description
30	0x0	LHS_OUTPUT_ENABLE: LCD_HSYNC pin output enable 0 = ENABLE 1 = DISABLE
28	0x0	LVS_OUTPUT_ENABLE: LVS pin output enable 0 = ENABLE 1 = DISABLE

Bit	Reset	Description
26	0x0	LSC1_OUTPUT_ENABLE: LCD_WR_N pin output enable 0 = ENABLE 1 = DISABLE
24	0x0	LSC0_OUTPUT_ENABLE: LCD_PCLK pin output enable 0 = ENABLE 1 = DISABLE
20	0x0	LPW2_OUTPUT_ENABLE: LCD_PWR2 pin output enable 0 = ENABLE 1 = DISABLE
18	0x0	LPW1_OUTPUT_ENABLE: LCD_PWR1 pin output enable 0 = ENABLE 1 = DISABLE
16	0x0	LPW0_OUTPUT_ENABLE: LCD_PWR0 pin output enable 0 = ENABLE 1 = DISABLE
2	0x0	LD17_OUTPUT_ENABLE: LD17 pin output enable 0 = ENABLE 1 = DISABLE
0	0x0	LD16_OUTPUT_ENABLE: LD16 pin output enable 0 = ENABLE 1 = DISABLE

29.8.5 DC_COM_PIN_OUTPUT_ENABLE2_0

Pin Output Enable 2

Offset: 304h | Read/Write: R/W | Reset: 0b1x1x0x1xxxxx0x1xxx0x1x0

Bit	Reset	Description
22	0x1	LPP_OUTPUT_ENABLE: LCD_D23 pin output enable 0 = ENABLE 1 = DISABLE (default after reset)
20	0x1	LDI_OUTPUT_ENABLE: LCD_D22 pin output enable 0 = ENABLE 1 = DISABLE (default after reset)
18	0x0	LM1_OUTPUT_ENABLE: LCD_M1 pin output enable 0 = ENABLE 1 = DISABLE
16	0x1	LM0_OUTPUT_ENABLE: LCD_CS1_N pin output enable 0 = ENABLE 1 = DISABLE
10	0x0	LVP1_OUTPUT_ENABLE: LCD_D20 pin output enable 0 = ENABLE 1 = DISABLE
8	0x1	LVP0_OUTPUT_ENABLE: LCD_DC1 pin output enable 0 = ENABLE 1 = DISABLE
4	0x0	LHP2_OUTPUT_ENABLE: LCD_D19 pin output enable 0 = ENABLE 1 = DISABLE

Bit	Reset	Description
2	0x1	LHP1_OUTPUT_ENABLE: LCD_D18 pin output enable 0 = ENABLE 1 = DISABLE
0	0x0	LHP0_OUTPUT_ENABLE: LCD_D21 pin output enable 0 = ENABLE 1 = DISABLE

29.8.6 DC_COM_PIN_OUTPUT_ENABLE3_0

Pin Output Enable 3

Offset: 305h | Read/Write: R/W | Reset: 0b1x1x1x1x1x1

Bit	Reset	Description
10	0x1	LSDI_OUTPUT_ENABLE: LSDI pin output enable 0 = ENABLE 1 = DISABLE (default after reset)
8	0x1	LSPI_OUTPUT_ENABLE: LSPI pin output enable 0 = ENABLE 1 = DISABLE (default after reset)
6	0x1	LDC_OUTPUT_ENABLE: LDC pin output enable 0 = ENABLE 1 = DISABLE (default after reset)
4	0x1	LCSN_OUTPUT_ENABLE: LCSN pin output enable 0 = ENABLE 1 = DISABLE (default after reset)
2	0x1	LSDA_OUTPUT_ENABLE: LSDA pin output enable 0 = ENABLE 1 = DISABLE (default after reset)
0	0x1	LSCK_OUTPUT_ENABLE: LSCK pin output enable 0 = ENABLE 1 = DISABLE (default after reset)

29.8.7 DC_COM_PIN_OUTPUT_POLARITY0_0

Pin Output Polarity 0

Pin Output Polarity registers

HIGH means the internal signal value is transmitted on pin as-is.

LOW means the internal signal is inverted before transmission on the pin.

Offset: 306h | Read/Write: R/W | Reset: 0b0x0x0x0x0x0x0x0x0x0x0x0x0x0x0x0x

Bit	Reset	Description
30	0x0	LD15_OUTPUT_POLARITY: LD15 pin output polarity 0 = HIGH 1 = LOW
28	0x0	LD14_OUTPUT_POLARITY: LD14 pin output polarity 0 = HIGH 1 = LOW

Bit	Reset	Description
26	0x0	LD13_OUTPUT_POLARITY: LD13 pin output polarity 0 = HIGH 1 = LOW
24	0x0	LD12_OUTPUT_POLARITY: LD12 pin output polarity 0 = HIGH 1 = LOW
22	0x0	LD11_OUTPUT_POLARITY: LD11 pin output polarity 0 = HIGH 1 = LOW
20	0x0	LD10_OUTPUT_POLARITY: LD10 pin output polarity 0 = HIGH 1 = LOW
18	0x0	LD9_OUTPUT_POLARITY: LD9 pin output polarity 0 = HIGH 1 = LOW
16	0x0	LD8_OUTPUT_POLARITY: LD8 pin output polarity 0 = HIGH 1 = LOW
14	0x0	LD7_OUTPUT_POLARITY: LD7 pin output polarity 0 = HIGH 1 = LOW
12	0x0	LD6_OUTPUT_POLARITY: LD6 pin output polarity 0 = HIGH 1 = LOW
10	0x0	LD5_OUTPUT_POLARITY: LD5 pin output polarity 0 = HIGH 1 = LOW
8	0x0	LD4_OUTPUT_POLARITY: LD4 pin output polarity 0 = HIGH 1 = LOW
6	0x0	LD3_OUTPUT_POLARITY: LD3 pin output polarity 0 = HIGH 1 = LOW
4	0x0	LD2_OUTPUT_POLARITY: LD2 pin output polarity 0 = HIGH 1 = LOW
2	0x0	LD1_OUTPUT_POLARITY: LD1 pin output polarity 0 = HIGH 1 = LOW
0	0x0	LD0_OUTPUT_POLARITY: LD0 pin output polarity 0 = HIGH 1 = LOW

29.8.8 DC_COM_PIN_OUTPUT_POLARITY1_0

Pin Output Polarity 1

Offset: 307h | Read/Write: R/W | Reset: 0b0x0x0x0xxx0x0x0xxxxxxxxxxxxx0x0

Bit	Reset	Description
30	0x0	LHS_OUTPUT_POLARITY: LCD_HSYNC pin output polarity 0 = HIGH 1 = LOW
28	0x0	LVS_OUTPUT_POLARITY: LVS pin output polarity 0 = HIGH 1 = LOW
26	0x0	LSC1_OUTPUT_POLARITY: LCD_WR_N pin output polarity 0 = HIGH 1 = LOW
24	0x0	LSC0_OUTPUT_POLARITY: LCD_PCLK pin output polarity. When used as PCLK (pixel clock), HIGH means that rising edge coincides with data transition, and LOW means that falling edge coincides with data transition. 0= active high 1= active low 0 = HIGH 1 = LOW
20	0x0	LPW2_OUTPUT_POLARITY: LCD_PWR2 pin output polarity 0 = HIGH 1 = LOW
18	0x0	LPW1_OUTPUT_POLARITY: LCD_PWR1 pin output polarity 0 = HIGH 1 = LOW
16	0x0	LPW0_OUTPUT_POLARITY: LCD_PWR0 pin output polarity 0 = HIGH 1 = LOW
2	0x0	LD17_OUTPUT_POLARITY: LD17 pin output polarity 0 = HIGH 1 = LOW
0	0x0	LD16_OUTPUT_POLARITY: LD16 pin output polarity 0 = HIGH 1 = LOW

29.8.9 DC_COM_PIN_OUTPUT_POLARITY2_0

Pin Output Polarity 2

Offset: 308h | Read/Write: R/W | Reset: 0b0x0x0x0xxxxx0x0xxx0x0x0

Bit	Reset	Description
22	0x0	LPP_OUTPUT_POLARITY: LCD_D23 pin output polarity 0 = HIGH 1 = LOW
20	0x0	LDI_OUTPUT_POLARITY: LCD_D22 pin output polarity 0 = HIGH 1 = LOW

Bit	Reset	Description
18	0x0	LM1_OUTPUT_POLARITY: LCD_M1 pin output polarity 0 = HIGH 1 = LOW
16	0x0	LM0_OUTPUT_POLARITY: LCD_CS1_N pin output polarity 0 = HIGH 1 = LOW
10	0x0	LVP1_OUTPUT_POLARITY: LCD_D20 pin output polarity 0 = HIGH 1 = LOW
8	0x0	LVP0_OUTPUT_POLARITY: LCD_DC1 pin output polarity 0 = HIGH 1 = LOW
4	0x0	LHP2_OUTPUT_POLARITY: LCD_D19 pin output polarity 0 = HIGH 1 = LOW
2	0x0	LHP1_OUTPUT_POLARITY: LCD_D18 pin output polarity 0 = HIGH 1 = LOW
0	0x0	LHP0_OUTPUT_POLARITY: LCD_D21 pin output polarity 0 = HIGH 1 = LOW

29.8.10 DC_COM_PIN_OUTPUT_POLARITY3_0

Pin Output Polarity 3

Offset: 309h | Read/Write: R/W | Reset: 0b0x0x0x0x0x0

Bit	Reset	Description
10	0x0	LSDI_OUTPUT_POLARITY: LSDI pin output polarity 0 = HIGH 1 = LOW
8	0x0	LSPI_OUTPUT_POLARITY: LSPI pin output polarity 0 = HIGH 1 = LOW
6	0x0	LDC_OUTPUT_POLARITY: LDC pin output polarity 0 = HIGH 1 = LOW
4	0x0	LCSN_OUTPUT_POLARITY: LCSN pin output polarity 0 = HIGH 1 = LOW
2	0x0	LSDA_OUTPUT_POLARITY: LSDA pin output polarity 0 = HIGH 1 = LOW
0	0x0	LSCK_OUTPUT_POLARITY: LSCK pin output polarity 0 = HIGH 1 = LOW

29.8.11 DC_COM_PIN_OUTPUT_DATA0_0

Pin Output Data 0

Pin Output data registers

Used for general purpose I/O, these values only take effect when OUTPUT_SELECT=1, typically.

The OUTPUT_DATA_MASK bits allow a subset of pins to be atomically changed without requiring a read/modify/write.

To change output data, the corresponding mask should be disabled (not masked).

Offset: 30ah | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	LD15_OUTPUT_DATA_MASK: LD15 pin output data mask 0 = MASKED 1 = NOTMASKED
30	0x0	LD15_OUTPUT_DATA: LD15 pin output data 0 = LOW 1 = HIGH
29	0x0	LD14_OUTPUT_DATA_MASK: LD14 pin output data mask 0 = MASKED 1 = NOTMASKED
28	0x0	LD14_OUTPUT_DATA: LD14 pin output data 0 = LOW 1 = HIGH
27	0x0	LD13_OUTPUT_DATA_MASK: LD13 pin output data mask 0 = MASKED 1 = NOTMASKED
26	0x0	LD13_OUTPUT_DATA: LD13 pin output data 0 = LOW 1 = HIGH
25	0x0	LD12_OUTPUT_DATA_MASK: LD12 pin output data 0 = MASKED 1 = NOTMASKED
24	0x0	LD12_OUTPUT_DATA: LD12 pin output data 0 = LOW 1 = HIGH
23	0x0	LD11_OUTPUT_DATA_MASK: LD11 pin output data mask 0 = MASKED 1 = NOTMASKED
22	0x0	LD11_OUTPUT_DATA: LD11 pin output data 0 = LOW 1 = HIGH
21	0x0	LD10_OUTPUT_DATA_MASK: LD10 pin output data mask 0 = MASKED 1 = NOTMASKED
20	0x0	LD10_OUTPUT_DATA: LD10 pin output data 0 = LOW 1 = HIGH

Bit	Reset	Description
19	0x0	LD9_OUTPUT_DATA_MASK: LD9 pin output data mask 0 = MASKED 1 = NOTMASKED
18	0x0	LD9_OUTPUT_DATA: LD9 pin output data 0 = LOW 1 = HIGH
17	0x0	LD8_OUTPUT_DATA_MASK: LD8 pin output data mask 0 = MASKED 1 = NOTMASKED
16	0x0	LD8_OUTPUT_DATA: LD8 pin output data 0 = LOW 1 = HIGH
15	0x0	LD7_OUTPUT_DATA_MASK: LD7 pin output data mask 0 = MASKED 1 = NOTMASKED
14	0x0	LD7_OUTPUT_DATA: LD7 pin output data 0 = LOW 1 = HIGH
13	0x0	LD6_OUTPUT_DATA_MASK: LD6 pin output data mask 0 = MASKED 1 = NOTMASKED
12	0x0	LD6_OUTPUT_DATA: LD6 pin output data 0 = LOW 1 = HIGH
11	0x0	LD5_OUTPUT_DATA_MASK: LD5 pin output data mask 0 = MASKED 1 = NOTMASKED
10	0x0	LD5_OUTPUT_DATA: LD5 pin output data 0 = LOW 1 = HIGH
9	0x0	LD4_OUTPUT_DATA_MASK: LD4 pin output data mask 0 = MASKED 1 = NOTMASKED
8	0x0	LD4_OUTPUT_DATA: LD4 pin output data 0 = LOW 1 = HIGH
7	0x0	LD3_OUTPUT_DATA_MASK: LD3 pin output data mask 0 = MASKED 1 = NOTMASKED
6	0x0	LD3_OUTPUT_DATA: LD3 pin output data 0 = LOW 1 = HIGH
5	0x0	LD2_OUTPUT_DATA_MASK: LD2 pin output data mask 0 = MASKED 1 = NOTMASKED
4	0x0	LD2_OUTPUT_DATA: LD2 pin output data 0 = LOW 1 = HIGH

Bit	Reset	Description
3	0x0	LD1_OUTPUT_DATA_MASK: LD1 pin output data mask 0 = MASKED 1 = NOTMASKED
2	0x0	LD1_OUTPUT_DATA: LD1 pin output data 0 = LOW 1 = HIGH
1	0x0	LD0_OUTPUT_DATA_MASK: LD0 pin output data mask 0 = MASKED 1 = NOTMASKED
0	0x0	LD0_OUTPUT_DATA: LD0 pin output data 0 = LOW 1 = HIGH

29.8.12 DC_COM_PIN_OUTPUT_DATA1_0

Pin Output Data 1

Offset: 30bh | Read/Write: R/W | Reset: 0b00000000xx000000xxxxxxxxxxxx0000

Bit	Reset	Description
31	0x0	LHS_OUTPUT_DATA_MASK: LCD_HSYNC pin output data mask 0 = MASKED 1 = NOTMASKED
30	0x0	LHS_OUTPUT_DATA: LCD_HSYNC pin output data 0 = LOW 1 = HIGH
29	0x0	LVS_OUTPUT_DATA_MASK: LVS pin output data mask 0 = MASKED 1 = NOTMASKED
28	0x0	LVS_OUTPUT_DATA: LVS pin output data 0 = LOW 1 = HIGH
27	0x0	LSC1_OUTPUT_DATA_MASK: LCD_WR_N pin output data mask 0 = MASKED 1 = NOTMASKED
26	0x0	LSC1_OUTPUT_DATA: LCD_WR_N pin output data 0 = LOW 1 = HIGH
25	0x0	LSC0_OUTPUT_DATA_MASK: LCD_PCLK pin output data mask 0 = MASKED 1 = NOTMASKED
24	0x0	LSC0_OUTPUT_DATA: LCD_PCLK pin output data 0 = LOW 1 = HIGH
21	0x0	LPW2_OUTPUT_DATA_MASK: LCD_PWR2 pin output data mask 0 = MASKED 1 = NOTMASKED
20	0x0	LPW2_OUTPUT_DATA: LCD_PWR2 pin output data 0 = LOW 1 = HIGH

Bit	Reset	Description
19	0x0	LPW1_OUTPUT_DATA_MASK: LCD_PWR1 pin output data mask 0 = MASKED 1 = NOTMASKED
18	0x0	LPW1_OUTPUT_DATA: LCD_PWR1 pin output data 0 = LOW 1 = HIGH
17	0x0	LPW0_OUTPUT_DATA_MASK: LCD_PWR0 pin output data mask 0 = MASKED 1 = NOTMASKED
16	0x0	LPW0_OUTPUT_DATA: LCD_PWR0 pin output data 0 = LOW 1 = HIGH
3	0x0	LD17_OUTPUT_DATA_MASK: LD17 pin output data mask 0 = MASKED 1 = NOTMASKED
2	0x0	LD17_OUTPUT_DATA: LD17 pin output data 0 = LOW 1 = HIGH
1	0x0	LD16_OUTPUT_DATA_MASK: LD16 pin output data mask 0 = MASKED 1 = NOTMASKED
0	0x0	LD16_OUTPUT_DATA: LD16 pin output data 0 = LOW 1 = HIGH

29.8.13 DC_COM_PIN_OUTPUT_DATA2_0

Pin Output Data 2

Offset: 30ch | Read/Write: R/W | Reset: 0b00000000xxxx0000xx000000

Bit	Reset	Description
23	0x0	LPP_OUTPUT_DATA_MASK: LCD_D23 pin output data mask 0 = MASKED 1 = NOTMASKED
22	0x0	LPP_OUTPUT_DATA: LCD_D23 pin output data 0 = LOW 1 = HIGH
21	0x0	LDI_OUTPUT_DATA_MASK: LCD_D22 pin output data mask 0 = MASKED 1 = NOTMASKED
20	0x0	LDI_OUTPUT_DATA: LCD_D22 pin output data 0 = LOW 1 = HIGH
19	0x0	LM1_OUTPUT_DATA_MASK: LCD_M1 pin output data mask 0 = MASKED 1 = NOTMASKED
18	0x0	LM1_OUTPUT_DATA: LCD_M1 pin output data 0 = LOW 1 = HIGH

Bit	Reset	Description
17	0x0	LM0_OUTPUT_DATA_MASK: LCD_CS1_N pin output data mask 0 = MASKED 1 = NOTMASKED
16	0x0	LM0_OUTPUT_DATA: LCD_CS1_N pin output data 0 = LOW 1 = HIGH
11	0x0	LVP1_OUTPUT_DATA_MASK: LCD_D20 pin output data mask 0 = MASKED 1 = NOTMASKED
10	0x0	LVP1_OUTPUT_DATA: LCD_D20 pin output data 0 = LOW 1 = HIGH
9	0x0	LVP0_OUTPUT_DATA_MASK: LCD_DC1 pin output data mask 0 = MASKED 1 = NOTMASKED
8	0x0	LVP0_OUTPUT_DATA: LCD_DC1 pin output data 0 = LOW 1 = HIGH
5	0x0	LHP2_OUTPUT_DATA_MASK: LCD_D19 pin output data mask 0 = MASKED 1 = NOTMASKED
4	0x0	LHP2_OUTPUT_DATA: LCD_D19 pin output data 0 = LOW 1 = HIGH
3	0x0	LHP1_OUTPUT_DATA_MASK: LCD_D18 pin output data mask 0 = MASKED 1 = NOTMASKED
2	0x0	LHP1_OUTPUT_DATA: LCD_D18 pin output data 0 = LOW 1 = HIGH
1	0x0	LHP0_OUTPUT_DATA_MASK: LCD_D21 pin output data mask 0 = MASKED 1 = NOTMASKED
0	0x0	LHP0_OUTPUT_DATA: LCD_D21 pin output data 0 = LOW 1 = HIGH

29.8.14 DC_COM_PIN_OUTPUT_DATA3_0

Pin Output Data 3

Pin Output data registers

Offset: 30dh | Read/Write: R/W | Reset: 0b000000000000

Bit	Reset	Description
11	0x0	LSDI_OUTPUT_DATA_MASK: LSDI pin output data mask 0 = MASKED 1 = NOTMASKED

Bit	Reset	Description
10	0x0	LSDI_OUTPUT_DATA: LSDI pin output data 0 = LOW 1 = HIGH
9	0x0	LSPI_OUTPUT_DATA_MASK: LSPI pin output data mask 0 = MASKED 1 = NOTMASKED
8	0x0	LSPI_OUTPUT_DATA: LSPI pin output data 0 = LOW 1 = HIGH
7	0x0	LDC_OUTPUT_DATA_MASK: LDC pin output data mask 0 = MASKED 1 = NOTMASKED
6	0x0	LDC_OUTPUT_DATA: LDC pin output data 0 = LOW 1 = HIGH
5	0x0	LCSN_OUTPUT_DATA_MASK: LCSN pin output data mask 0 = MASKED 1 = NOTMASKED
4	0x0	LCSN_OUTPUT_DATA: LCSN pin output data 0 = LOW 1 = HIGH
3	0x0	LSDA_OUTPUT_DATA_MASK: LSDA pin output data mask 0 = MASKED 1 = NOTMASKED
2	0x0	LSDA_OUTPUT_DATA: LSDA pin output data 0 = LOW 1 = HIGH
1	0x0	LSCK_OUTPUT_DATA_MASK: LSCK pin output data mask 0 = MASKED 1 = NOTMASKED
0	0x0	LSCK_OUTPUT_DATA: LSCK pin output data 0 = LOW 1 = HIGH

29.8.15 DC_COM_PIN_INPUT_ENABLE0_0

Pin Input Enable 0

Pin Input Enable registers

Offset: 30eh | Read/Write: R/W | Reset: 0b0x0x0x0x0x0x0x0x0x0x0x0x0x0x0x0x

Bit	Reset	Description
30	0x0	LD15_INPUT_ENABLE: LD15 pin input enable 0 = DISABLE 1 = ENABLE
28	0x0	LD14_INPUT_ENABLE: LD14 pin input enable 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
26	0x0	LD13_INPUT_ENABLE: LD13 pin input enable 0 = DISABLE 1 = ENABLE
24	0x0	LD12_INPUT_ENABLE: LD12 pin input enable 0 = DISABLE 1 = ENABLE
22	0x0	LD11_INPUT_ENABLE: LD11 pin input enable 0 = DISABLE 1 = ENABLE
20	0x0	LD10_INPUT_ENABLE: LD10 pin input enable 0 = DISABLE 1 = ENABLE
18	0x0	LD9_INPUT_ENABLE: LD9 pin input enable 0 = DISABLE 1 = ENABLE
16	0x0	LD8_INPUT_ENABLE: LD8 pin input enable 0 = DISABLE 1 = ENABLE
14	0x0	LD7_INPUT_ENABLE: LD7 pin input enable 0 = DISABLE 1 = ENABLE
12	0x0	LD6_INPUT_ENABLE: LD6 pin input enable 0 = DISABLE 1 = ENABLE
10	0x0	LD5_INPUT_ENABLE: LD5 pin input enable 0 = DISABLE 1 = ENABLE
8	0x0	LD4_INPUT_ENABLE: LD4 pin input enable 0 = DISABLE 1 = ENABLE
6	0x0	LD3_INPUT_ENABLE: LD3 pin input enable 0 = DISABLE 1 = ENABLE
4	0x0	LD2_INPUT_ENABLE: LD2 pin input enable 0 = DISABLE 1 = ENABLE
2	0x0	LD1_INPUT_ENABLE: LD1 pin input enable 0 = DISABLE 1 = ENABLE
0	0x0	LD0_INPUT_ENABLE: LD0 pin input enable 0 = DISABLE 1 = ENABLE

29.8.16 DC_COM_PIN_INPUT_ENABLE1_0

Pin Input Enable 1

Offset: 30fh | Read/Write: R/W | Reset: 0b0x0x0x0xxx0x0x0xxxxxxxxxxxxx0x0

Bit	Reset	Description

Bit	Reset	Description
30	0x0	LHS_INPUT_ENABLE: LCD_HSYNC pin input enable 0 = DISABLE 1 = ENABLE
28	0x0	LVS_INPUT_ENABLE: LCD_VSYNC pin input enable 0 = DISABLE 1 = ENABLE
26	0x0	LSC1_INPUT_ENABLE: LCD_WR_N pin input enable 0 = DISABLE 1 = ENABLE
24	0x0	LSC0_INPUT_ENABLE: LCD_PCLK pin input enable 0 = DISABLE 1 = ENABLE
20	0x0	LPW2_INPUT_ENABLE: LCD_PWR2 pin input enable 0 = DISABLE 1 = ENABLE
18	0x0	LPW1_INPUT_ENABLE: LCD_PWR1 pin input enable 0 = DISABLE 1 = ENABLE
16	0x0	LPW0_INPUT_ENABLE: LCD_PWR0 pin input enable 0 = DISABLE 1 = ENABLE
2	0x0	LD17_INPUT_ENABLE: LD17 pin input enable 0 = DISABLE 1 = ENABLE
0	0x0	LD16_INPUT_ENABLE: LD16 pin input enable 0 = DISABLE 1 = ENABLE

29.8.17 DC_COM_PIN_INPUT_ENABLE2_0

Pin Input Enable 2

Offset: 310h | Read/Write: R/W | Reset: 0b0x0x0x0xxxxx0x0xxx0x0x0

Bit	Reset	Description
22	0x0	LPP_INPUT_ENABLE: LCD_D23 pin input enable 0 = DISABLE 1 = ENABLE
20	0x0	LDI_INPUT_ENABLE: LCD_D22 pin input enable 0 = DISABLE 1 = ENABLE
18	0x0	LM1_INPUT_ENABLE: LCD_M1 pin input enable 0 = DISABLE 1 = ENABLE
16	0x0	LM0_INPUT_ENABLE: LCD_CS1_N pin input enable 0 = DISABLE 1 = ENABLE
10	0x0	LVP1_INPUT_ENABLE: LCD_D20 pin input enable 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	0x0	LVP0_INPUT_ENABLE: LCD_DC1 pin input enable 0 = DISABLE 1 = ENABLE
4	0x0	LHP2_INPUT_ENABLE: LCD_D19 pin input enable 0 = DISABLE 1 = ENABLE
2	0x0	LHP1_INPUT_ENABLE: LCD_D18 pin input enable 0 = DISABLE 1 = ENABLE
0	0x0	LHP0_INPUT_ENABLE: LCD_D21 pin input enable 0 = DISABLE 1 = ENABLE

29.8.18 DC_COM_PIN_INPUT_ENABLE3_0

Pin Input Enable 3

Offset: 311h | Read/Write: R/W | Reset: 0b0x0x0x0x0x0

Bit	Reset	Description
10	0x0	LSDI_INPUT_ENABLE: LSDI pin input enable 0 = DISABLE 1 = ENABLE
8	0x0	LSPI_INPUT_ENABLE: LSPI pin input enable 0 = DISABLE 1 = ENABLE
6	0x0	LDC_INPUT_ENABLE: LDC pin input enable 0 = DISABLE 1 = ENABLE
4	0x0	LCSN_INPUT_ENABLE: LCSN pin input enable 0 = DISABLE 1 = ENABLE
2	0x0	LSDA_INPUT_ENABLE: LSDA pin input enable 0 = DISABLE 1 = ENABLE
0	0x0	LSCK_INPUT_ENABLE: LSCK pin input enable 0 = DISABLE 1 = ENABLE

29.8.19 DC_COM_PIN_INPUT_DATA0_0

Pin Input Data 0

Pin Input Data registers (read-only)

Offset: 312h | Read/Write: RO | Reset: 0b000000000000000000

Bit	Reset	Description
17	0x0	LD17_INPUT_DATA: LD17 pin input data
16	0x0	LD16_INPUT_DATA: LD16 pin input data

Bit	Reset	Description
15	0x0	LD15_INPUT_DATA: LD15 pin input data
14	0x0	LD14_INPUT_DATA: LD14 pin input data
13	0x0	LD13_INPUT_DATA: LD13 pin input data
12	0x0	LD12_INPUT_DATA: LD12 pin input data
11	0x0	LD11_INPUT_DATA: LD11 pin input data
10	0x0	LD10_INPUT_DATA: LD10 pin input data
9	0x0	LD9_INPUT_DATA: LD9 pin input data
8	0x0	LD8_INPUT_DATA: LD8 pin input data
7	0x0	LD7_INPUT_DATA: LD7 pin input data
6	0x0	LD6_INPUT_DATA: LD6 pin input data
5	0x0	LD5_INPUT_DATA: LD5 pin input data
4	0x0	LD4_INPUT_DATA: LD4 pin input data
3	0x0	LD3_INPUT_DATA: LD3 pin input data
2	0x0	LD2_INPUT_DATA: LD2 pin input data
1	0x0	LD1_INPUT_DATA: LD1 pin input data
0	0x0	LD0_INPUT_DATA: LD0 pin input data

29.8.20 DC_COM_PIN_INPUT_DATA1_0

Pin Input Data 1

Offset: 313h | Read/Write: RO | Reset: 0x0000000

Bit	Reset	Description
25	0x0	LSDI_INPUT_DATA: LSDI pin input data
24	0x0	LSPI_INPUT_DATA: LSPI pin input data
23	0x0	LDC_INPUT_DATA: LDC pin input data
22	0x0	LCSN_INPUT_DATA: LCSN pin input data
21	0x0	LSDA_INPUT_DATA: LSDA pin input data
20	0x0	LSCK_INPUT_DATA: LSCK pin input data
19	0x0	LPP_INPUT_DATA: LPP pin input data
18	0x0	LDI_INPUT_DATA: LDI pin input data
17	0x0	LM1_INPUT_DATA: LM1 pin input data
16	0x0	LM0_INPUT_DATA: LM0 pin input data

Bit	Reset	Description
13	0x0	LVP1_INPUT_DATA: LVP1 pin input data
12	0x0	LVP0_INPUT_DATA: LVP0 pin input data
10	0x0	LHP2_INPUT_DATA: LHP2 pin input data
9	0x0	LHP1_INPUT_DATA: LHP1 pin input data
8	0x0	LHP0_INPUT_DATA: LHP0 pin input data
7	0x0	LHS_INPUT_DATA: LHS pin input data
6	0x0	LVS_INPUT_DATA: LVS pin input data
5	0x0	LSC1_INPUT_DATA: LSC1 pin input data
4	0x0	LSC0_INPUT_DATA: LSC0 pin input data
2	0x0	LPW2_INPUT_DATA: LPW2 pin input data
1	0x0	LPW1_INPUT_DATA: LPW1 pin input data
0	0x0	LPW0_INPUT_DATA: LPW0 pin input data

29.8.21 DC_COM_PIN_OUTPUT_SELECT0_0

3 bits are used to select output on each pin and they are defined as follows:

Pad Name	Pin Output Select							
	0 Output Signal	1 Output Signal	2 Output Signal	3 Output Signal	4 Output Signal	5 Output Signal	6 Output Signal	7 Output Signal
LD17	LD17	LD17 Out	LPD17	0	0	0	0	0
LD16	LD16	LD16 Out	LPD16	0	0	0	0	0
LD15	LD15	LD15 Out	LPD15	0	0	0	0	0
LD14	LD14	LD14 Out	LPD14	0	0	0	0	0
LD13	LD13	LD13 Out	LPD13	0	0	0	0	0
LD12	LD12	LD12 Out	LPD12	0	0	0	0	0
LD11	LD11	LD11 Out	LPD11	0	0	0	0	0
LD10	LD10	LD10 Out	LPD10	0	SD2	0	0	0
LD9	LD9	LD9 Out	LPD9	0	SD2_	0	0	0
LD8	LD8	LD8 Out	LPD8	0	STP	0	0	0
LD7	LD7	LD7 Out	LPD7	0	SDT	0	0	0
LD6	LD6	LD6 Out	LPD6	0	STH	0	0	0
LD5	LD5	LD5 Out	LPD5	0	SD1	0	0	0
LD4	LD4	LD4 Out	LPD4	0	SD1_	0	0	0
LD3	LD3	LD3 Out	LPD3	0	SD0	0	0	0
LD2	LD2	LD2 Out	LPD2	0	SD0_	0	0	0
LD1	LD1	LD1 Out	LPD1	0	SC	0	0	0
LD0	LD0	LD0 Out	LPD0	0	SC_	0	0	0
LPW0	PW0	LPW0 Out	PW1	PM0	PW2	MD0	LPD4	LSDA
LPW1	PW1	LPW1 Out	PW2	PM1	PW3	MD1	LPD8	PW4
LPW2	PW2	LPW2 Out	PW3	PM0	PW4	MD2	LPD5	PW1

Pad Name	Pin Output Select							
	0 Output Signal	1 Output Signal	2 Output Signal	3 Output Signal	4 Output Signal	5 Output Signal	6 Output Signal	7 Output Signal
LSC0	SC0	LSC0 Out	0	0	0	0	0	0
LSC1	SC1	LSC1 Out	DE	0	0	0	0	LSCK
LVS	Vsync	LVS Out	0	PM1	0	MD3	0	0
LHS	Hsync	LHS Out	0	PM0	0	MD2	0	0
LHP0	H Pulse 0	LHP0 Out	LD21	PM0	0	MD0	LPD7	0
LHP1	H Pulse 1	LHP1 Out	LD18	PM1	0	MD1	LPD0	0
LHP2	H Pulse 2	LHP2 Out	LD19	PM0	V Pulse 2	MD2	Hsync	0
LVP0	V Pulse 0	LVP0 Out	0	PM0	0	MD3	V Pulse1	LDC
LVP1	V Pulse 1	LVP1 Out	LD20	PM1	PW4	MD3	LPD6	0
LM0	M0	LM0 Out	H Pulse 0	PM0	V Pulse 2	MD0	DE	SCS_
LM1	M1	LM1 Out	LD21	PM1	V Pulse 3	MD1	LPD1	0
LDI	DI	LDI Out	LD22	PM0	SCS_	MD2	LPD2	0
LPP	PP	LPP Out	LD23	PM1	V Pulse 3	MD3	LPD3	0
LSCK	SCK	LSCK Out	0	PM0	0	MD0	LPD3	0
LSDA	SDA	LSDA Out	SCS_	PM1	0	MD1	LPD4	0
LCS_	SCS_	LCS_ Out	LD22	PM0	DE	MD2	LPD5	0
LDC	SDC	LDC Out	Vsync	PM1	V Pulse 1	MD3	LPD6	0
LSPI	SPI busy	LSPI Out	DE	PM0	DC Clk	MD0	0	0
LSDI	SDI	LSDI Out	0	PM1	0	MD1	0	0

Notes:
1. LD[23-0] contain pixel data for 1-pixel/1-clock parallel interface
2. LPD[17-0] contain pixel data for non 1-pixel/1-clock parallel interface
3. If output select is set to 1, then corresponding Pin Output Data register value is output (pin is used as general purpose output).
4. If output select is set to 1 AND display is in 2p1c18 (aka notebook) mode, then all outputs are overloaded. NB this doesn't apply to lsc1 which must be programmed to output-select==0 to get the clock signal.
5. For dual display, each display has its own instance of the above pad macro. Set up each display according to the panel interface and insure that pins are used by only one display.

A set of registers outside of display, PIN_MUX_CTL_**, control which one out of 4 inputs, including display and displayb, go to the pads.

Pin Output Select 0

Offset: 314h | Read/Write: R/W | Reset: 0b000x000x000x000x000x000x000x000

Bit	Reset	Description
30:28	0x0	LD7_OUTPUT_SELECT: LD7 pin output select
26:24	0x0	LD6_OUTPUT_SELECT: LD6 pin output select
22:20	0x0	LD5_OUTPUT_SELECT: LD5 pin output select
18:16	0x0	LD4_OUTPUT_SELECT: LD4 pin output select
14:12	0x0	LD3_OUTPUT_SELECT: LD3 pin output select
10:8	0x0	LD2_OUTPUT_SELECT: LD2 pin output select

Bit	Reset	Description
6:4	0x0	LD1_OUTPUT_SELECT: LD1 pin output select
2:0	0x0	LD0_OUTPUT_SELECT: LD0 pin output select

29.8.22 DC_COM_PIN_OUTPUT_SELECT1_0

Pin Output Select 1

Offset: 315h | Read/Write: R/W | Reset: 0b000x000x000x000x000x000x000x000

Bit	Reset	Description
30:28	0x0	LD15_OUTPUT_SELECT: LD15 pin output select
26:24	0x0	LD14_OUTPUT_SELECT: LD14 pin output select
22:20	0x0	LD13_OUTPUT_SELECT: LD13 pin output select
18:16	0x0	LD12_OUTPUT_SELECT: LD12 pin output select
14:12	0x0	LD11_OUTPUT_SELECT: LD11 pin output select
10:8	0x0	LD10_OUTPUT_SELECT: LD10 pin output select
6:4	0x0	LD9_OUTPUT_SELECT: LD9 pin output select
2:0	0x0	LD8_OUTPUT_SELECT: LD8 pin output select

29.8.23 DC_COM_PIN_OUTPUT_SELECT2_0

Pin Output Select 2

Offset: 316h | Read/Write: R/W | Reset: 0b000x000

Bit	Reset	Description
6:4	0x0	LD17_OUTPUT_SELECT: LD17 pin output select
2:0	0x0	LD16_OUTPUT_SELECT: LD16 pin output select

29.8.24 DC_COM_PIN_OUTPUT_SELECT3_0

Pin Output Select 3

Offset: 317h | Read/Write: R/W | Reset: 0b000x000x000x000xxxxx000x000x000

Bit	Reset	Description
30:28	0x0	LHS_OUTPUT_SELECT: LCD_HSYNC pin output select
26:24	0x0	LVS_OUTPUT_SELECT: LCD_VSYNC pin output select
22:20	0x0	LSC1_OUTPUT_SELECT: LCD_WR_N pin output select
18:16	0x0	LSC0_OUTPUT_SELECT: LCD_PCLK pin output select
10:8	0x0	LPW2_OUTPUT_SELECT: LCD_PWR2 pin output select

Bit	Reset	Description
6:4	0x0	LPW1_OUTPUT_SELECT: LCD_PWR1 pin output select
2:0	0x0	LPW0_OUTPUT_SELECT: LCD_PWR0 pin output select

29.8.25 DC_COM_PIN_OUTPUT_SELECT4_0

Pin Output Select 4

Offset: 318h | Read/Write: R/W | Reset: 0b000x000xxxxx000x000x000

Bit	Reset	Description
22:20	0x0	LVP1_OUTPUT_SELECT: LCD_D20 pin output select
18:16	0x0	LVP0_OUTPUT_SELECT: LCD_DC1 pin output select
10:8	0x0	LHP2_OUTPUT_SELECT: LCD_D19 pin output select
6:4	0x0	LHP1_OUTPUT_SELECT: LCD_D18 pin output select
2:0	0x0	LHP0_OUTPUT_SELECT: LCD_D21 pin output select

29.8.26 DC_COM_PIN_OUTPUT_SELECT5_0

Pin Output Select 5

Offset: 319h | Read/Write: R/W | Reset: 0b000x000x000x000

Bit	Reset	Description
14:12	0x0	LPP_OUTPUT_SELECT: LCD_D23 pin output select
10:8	0x0	LDI_OUTPUT_SELECT: LCD_D22 pin output select
6:4	0x0	LM1_OUTPUT_SELECT: LCD_M1 pin output select
2:0	0x0	LM0_OUTPUT_SELECT: LCD_CS1_N pin output select

29.8.27 DC_COM_PIN_OUTPUT_SELECT6_0

Pin Output Select 6

Offset: 31ah | Read/Write: R/W | Reset: 0b000x000x000x000x000x000

Bit	Reset	Description
22:20	0x0	LSDI_OUTPUT_SELECT: LSDI pin output select
18:16	0x0	LSPI_OUTPUT_SELECT: LSPI pin output select
14:12	0x0	LDC_OUTPUT_SELECT: LDC pin output select
10:8	0x0	LCSN_OUTPUT_SELECT: LCSN pin output select
6:4	0x0	LSDA_OUTPUT_SELECT: LSDA pin output select
2:0	0x0	LSCK_OUTPUT_SELECT: LSCK pin output select

29.8.28 DC_COM_PIN_MISC_CONTROL_0

Pin Miscellaneous Control

Pin Miscellaneous Control

Offset: 31bh | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
2	0x0	DISP_CLOCK_OUTPUT: Display Clock (DCLK) Enable 0= disable 1= enable display clock to be output on LCD_DE pin (LCD_DE output select must be appropriately programmed for this to be effective) 0 = DISABLE 1 = ENABLE

29.8.29 DC_COM_PM0_CONTROL_0

PM0 signal Control

Class: Pulse Width Modulation

PM0 signal is programmable pulse width modulation signal that can be output on several pins. The control register should be initialized once before PM0 is enabled.

Note: The period is expressed as multiples of 4 times the divider value.
The actual period - in clock cycles - is given by:
period = (1 + PM0_CLOCK_DIVIDER) * ((PM0_PERIOD + 1) * 4)

Offset: 31ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:18	X	PM0_PERIOD: PM0 Period (4, 8, ... , 256 clock cycles)
9:4	X	PM0_CLOCK_DIVIDER: PM0 Clock Divider (1 to 64)
1:0	X	PM0_CLOCK_SELECT: PM0 Clock Select 0= output of shift clock divider 1= pixel clock 2= line clock 3= frame clock Notes: 1) Pixel clock, line clock, and frame clock is running only when PW0 signal is enabled. 2) In non-continuous mode, shift clock and pixel clock run continuously, but line clock and frame clock only run while a frame is being sent.

29.8.30 DC_COM_PM0_DUTY_CYCLE_0

PM0 Duty Cycle

A counter repeatedly counts up from 0 to ((PM0_PERIOD << 2) + 3) pre-scaled cycles.

The period always starts with the output value == 1. After DUTY_CYCLE number of pre-scaled cycles, the output value is cleared for the remainder of the period. In order to output constant 0, simply set the DUTY_CYCLE to 0. To output a constant 1, set DUTY_CYCLE to be any value greater than ((PM0_PERIOD << 2) + 3).

Offset: 31dh | Read/Write: R/W | Reset: 0bxxxxxxxxx

Bit	Reset	Description
8:0	X	PM0_DUTY_CYCLE: PM0 Duty Cycle (or D) From 1/P to D/P pulse high time where P is the period. This must not be larger than the period.

29.8.31 DC_COM_PM1_CONTROL_0

PM1 signal Control

PM1 signal is programmable pulse width modulation signal that can be output on several pins.

The control register should be initialized once before PM1 is enabled.

The actual period (in clock cycles) is given by:

PERIOD = (1 + PM1_CLOCK_DIVIDER) * ((PM1_PERIOD + 1) * 4)

Offset: 31eh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:18	X	PM1_PERIOD: PM1 Period (4, 8, ... , 256 clock cycles)
9:4	X	PM1_CLOCK_DIVIDER: PM1 Clock Divider (1 to 64)
1:0	X	PM1_CLOCK_SELECT: PM1 Clock Select 0= output of shift clock divider 1= pixel clock 2= line clock 3= frame clock Notes: 1) Pixel clock, line clock, and frame clock is running only when PW0 signal is enabled. 2) In non-continuous mode, shift clock and pixel clock run continuously, but line clock and frame clock only run while a frame is being sent.

29.8.32 DC_COM_PM1_DUTY_CYCLE_0

PM1 Duty Cycle

A counter repeatedly counts up from 0 to ((PM1_PERIOD << 2) + 3) pre-scaled cycles.

The period always starts with the output value == 1. After DUTY_CYCLE number of pre-scaled cycles, the output value is cleared for the remainder of the period. In order to output constant 0, simply set the DUTY_CYCLE to 0. To output a constant 1, set DUTY_CYCLE to be any value greater than ((PM1_PERIOD << 2) + 3).

Offset: 31fh | Read/Write: R/W | Reset: 0bxxxxxxxxx

Bit	Reset	Description
8:0	X	PM1_DUTY_CYCLE: PM1 Duty Cycle from 1/P to P/P pulse high time where P is the period. This must not be larger than the period.

29.8.33 DC_COM_SPI_CONTROL_0

SPI Control

Class: Serial Peripheral Interface (SPI)

SPI interface is a 3-pin or 4-pin serial interface which is typically used to program registers in display device. However, for display with built-in frame buffer, it can also be used to write pixel data to the built-in frame buffer. Currently only write cycles are supported.

LCD SPI interface signal consists of:

- SPI Clock (SCK) which can be output on LCD_SCK pin.
- SPI Data (SDA) which can be output on LCD_SDOUT pin.
- Optional SPI Data/Command (SDC) which can be output on LCD_DC0 pin.
- Main-Display SPI Chip Select (Main SCS_) signal which can be output on LCS_ pin
- Sub-Display SPI Chip Select (Sub SCS_) signal which can be output on LCD_CS1_N or LCD_D22 or LCD_SDOUT pins - this is only used only if there is a sub display

Theoretically if two LCD panels (main and sub displays) are connected, then SPI interface can be connected to both LCD panels and shared between the two panels.

In this case two Chip Select pins are required and LCS_ should be connected to the main display and other pin has to be selected to output the sub display SCS_.

Internally two SPI chip select signals (lmscs_ and lsscs_ for main and sub display SPI chip select) are generated.

When SPI is enabled, there are three possible SPI transactions:

- SPI write can be triggered by host by writing to HSPI_CS_DC register
- SPI write can be triggered to send pixel data from frame buffer
- SPI write can be triggered to send Initialization Sequence (IS). Frame initialization sequence is sent one line prior to active display line.

For LCD SPI, pixel data can only be sent to either Main-Display or Sub-Display but not to both but host SPI and IS SPI can be sent to both Main and Sub displays simultaneously.

In case a. and c. the SPI root clock is derived from output of shift clock divider which is then further divided by SPI_CLK_DIVIDER. For both cases, the SPI number of bits/cycle is determined by SPI_BITS_PER_CYCLE and the SPI data direction is determined by SPI_DATA_DIRECTION.

In case b. the SPI root clock is derived from output of shift clock divider with no further division. For this case, the SPI number of bits/cycle is determined by correct programming of pixel clock divider.

If case a. is enabled at the same time with case b. and c. then host SPI triggers is delayed to the beginning of horizontal display active time of a line that does not have IS SPI or LCD SPI cycles. This means that if all cases are enabled, the vertical blank time must be at least 2 lines otherwise host SPI cycles cannot be executed.

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

Bit	Reset	Description
25:24	X	SPI_STATUS_ENABLE: SPI Status Enable 00= SPI status disable 01= SPI status enabled for host SPI only 10= SPI status enabled for IS and LCD SPI only 11= SPI status enabled for all SPI cycles SPI status is reflected in SPI_BUSY bit and can generate interrupt. SPI status indicates when SPI module is busy (SPI write cycles are in progress) so its falling edge should be used to generate interrupt. SPI status can also be output on LCD_DE pin. When Host SPI is triggered, the SPI busy is asserted within three display clock cycles after the end of the host write cycle. SPI status is disabled when SPI_ENABLE bit is disabled.
20:16	X	SPI_CLK_DIVIDER: SPI Clock Divider (1 to 32) This clock divider is used only if SPI is enabled for host writes (Host SPI) or for sending initialization sequence (IS SPI). Programmed value is 1 less than the desired (actual) clock divider value. This parameter is forced to 0 (clock divide by 1) for LCD SPI.
7:4	X	SPI_BITS_PER_CYCLE: SPI Bits per Cycle This is valid for Host and IS SPI only. This parameter determines the number of bits/cycle when SPI is used for host write or for sending initialization sequence. If SPI is used for sending pixel data to the display then pixel clock divider determines the SPI bits/cycle. SPI8DC is 8-bit SPI plus data/command bit SPI16DC is 16-bit SPI plus data/command bit SPI16SB is 16-bit SPI plus an 8-bit start byte preceding the 16-bit data. 0 = SPI8 1 = SPI8DC 2 = SPI12 3 = SPI16 4 = SPI16DC 5 = SPI16SB 6 = SPI18 7 = SPI24
3	X	SPI_DATA_DIRECTION: SPI Data Direction. This is valid for Host SPI and for sending initialization sequence (IS SPI) only. Note that data direction does not affect the start byte direction (always msb to lsb) and position (always first 8-bit of serial data) for SPI16SB mode. 0 = MSB2LSB 1 = LSB2MSB
1:0	X	SPI_SERIAL_CLK_CONTROL: SPI Serial Clock Control 0= SCK rising edge is active edge 1-clock chip select and no SCK clock edge to latch chip select 1= SCK rising edge is active edge 2-clock chip select with SCK rising clock edge to latch it 2= SCK falling edge is active edge 1-clock chip select and no SCK clock edge to latch chip select 3= SCK falling edge is active edge 2-clock chip select with SCK falling clock edge to latch it This is valid for Host, IS, and LCD SPI

29.8.34 DC_COM_SPI_START_BYTE_0

SPI Start Byte

SPI Start Bytes are used only for SPI16SB mode (start byte plus 16-bit data). Data direction does not affect the start byte direction (always msb to lsb) and position (always first 8-bit of serial data).

Offset: 321h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:8	X	SPI_COMMAND_START_BYTE: SPI Command Start Byte This is valid for Host, IS, and LCD SPI.
7:0	X	SPI_DATA_START_BYTE: SPI Data Start Byte This is valid for Host, IS, and LCD SPI.

29.8.35 DC_COM_HSPI_WRITE_DATA_AB_0

Host SPI Write Data A & B

These registers are used to send write data for Host SPI write cycles (HSPI_ENABLE = 1).

For the tables below, these abbreviations of the registers are used:

- HSPIWDA = HSPI_WRITE_DATA_A
- HSPIWDB = HSPI_WRITE_DATA_B
- HSPIWDC = HSPI_WRITE_DATA_C
- HSPIWDD = HSPI_WRITE_DATA_D
- HSPIMCS = HSPI_MAIN_CS
- HSPISCS = HSPI_SUB_CS
- HSPIDC = HSPI_DC

For 8-bit SPI, up to 4 write cycles can be performed (SPI_BITS_PER_CYCLE equals SPI8)

Write cycle	**0**	**1**	**2**	**3**
Data	HSPIWDA[7-0]	HSPIWDA[15-8]	HSPIWDB[7-0]	HSPIWDB[15-8]
Main SCS_	HSPIMCS[0]	HSPIMCS[1]	HSPIMCS[2]	HSPIMCS[3]
Sub SCS_	HSPISCS[0]	HSPISCS[1]	HSPISCS[2]	HSPICSC[3]
SDC	HSPIDC[0]	HSPIDC[1]	HSPIDC[2]	HSPIDC[3]

For SPI8DC mode, a 9-bit SPI, the first data bit sent comes from HSPI_DC register and the other eight bits come as shown in the table below. Up to 4 write cycles can be performed.

Write cycle	**0**	**1**
Data	HSPIDC[0], HSPIWDA[7-0]	HSPIDC[1], HSPIWDA[15-8]
Main SCS_	HSPIMCS[0]	HSPIMCS[1]
Sub SCS_	HSPISCS[0]	HSPISCS[1]
SDC	HSPIDC[0]	HSPIDC[1]
Write cycle	**2**	**3**
Data	HSPIDC[2], HSPIWDB[7-0]	HSPIDC[3], HSPIWDB[15-8]
Main SCS_	HSPIMCS[2]	HSPIMCS[3]
Sub SCS_	HSPISCS[2]	HSPICSC[3]
SDC	HSPIDC[2]	HSPIDC[3]

For 12-bit SPI, up to 4 write cycles can be performed (SPI12)

Write cycle	0	1	2	3
Data	HSPIWDA[11-0]	HSPIWDB[11-0]	HSPIWDC[11-0]	HSPIWDD[11-0]
Main SCS_	HSPIMCS[0]	HSPIMCS[1]	HSPIMCS[2]	HSPIMCS[3]
Sub SCS_	HSPISCS[0]	HSPISCS[1]	HSPISCS[2]	HSPISCS[3]
SDC	HSPIDC[0]	HSPIDC[1]	HSPIDC[2]	HSPIDC[3]

For 16-bit SPI, up to 4 write cycles can be performed (SPI16, SPI16SB). Note for SPI16SB mode, the first write cycle will be the start byte, followed by write cycle 0, as shown in the table below.

Write cycle	0	1	2	3
Data	HSPIWDA[15-0]	HSPIWDB[15-0]	HSPIWDC[15-0]	HSPIWDD[15-0]
Main SCS_	HSPIMCS[0]	HSPIMCS[1]	HSPIMCS[2]	HSPIMCS[3]
Sub SCS_	HSPISCS[0]	HSPISCS[1]	HSPISCS[2]	HSPISCS[3]
SDC	HSPIDC[0]	HSPIDC[1]	HSPIDC[2]	HSPIDC[3]

For SPI16DC mode, a 17-bit SPI, the first data bit sent comes from HSPI_DC register and the other sixteen bits come as shown in the table below. Up to 4 write cycles can be performed.

Write cycle	0	1
Data	HSPIDC[0], HSPIWDA[15-0]	HSPIDC[1], HSPIWDB[15-0]
Main SCS_	HSPIMCS[0]	HSPIMCS[1]
Sub SCS_	HSPISCS[0]	HSPISCS[1]
SDC	HSPIDC[0]	HSPIDC[1]
Write cycle	**2**	**3**
Data	HSPIDC[2], HSPIWDC[15-0]	HSPIDC[3], HSPIWDD[15-0]
Main SCS_	HSPIMCS[2]	HSPIMCS[3]
Sub SCS_	HSPISCS[2]	HSPISCS[3]
SDC	HSPIDC[2]	HSPIDC[3]

For 18-bit SPI, up to 2 write cycles can be performed (SPI18)

Write cycle	0	1
Data	HSPIWDB[1-0], HSPIWDA[15-0]	HSPIWDC[1-0], HSPIWDD[15-0]
Main SCS_	HSPIMCS[0]	HSPIMCS[1]
Sub SCS_	HSPISCS[0]	HSPISCS[1]
SDC	HSPIDC[0]	HSPIDC[1]

For 24-bit SPI, up to 2 write cycles can be performed (SPI24)

Write cycle	0	1
Data	HSPIWDB[7-0], HSPIWDA[15-0]	HSPIWDC[7-0], HSPIWDD[15-0]
Main SCS_	HSPIMCS[0]	HSPIMCS[1]
Sub SCS_	HSPISCS[0]	HSPISCS[1]
SDC	HSPIDC[0]	HSPIDC[1]

Offset: 322h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	HSPI_WRITE_DATA_B: Host SPI Write Data B bits 15-0
15:0	X	HSPI_WRITE_DATA_A: Host SPI Write Data A bits 15-0

29.8.36 DC_COM_HSPI_WRITE_DATA_CD_0

Host SPI Write Data C & D

Offset: 323h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	HSPI_WRITE_DATA_D: Host SPI Write Data D bits 15-0
15:0	X	HSPI_WRITE_DATA_C: Host SPI Write Data C bits 15-0

29.8.37 DC_COM_HSPI_CS_DC_0

Host SPI Chip Select and Data/Command

A write to this register will trigger the Host SPI write cycle if SPI_ENABLE and HSPI_ENABLE are both enabled. Writing to this register with HSPI_RAISE enabled will cause the raise vector to be returned after all the host SPI cycles are completed. Up to four host SPI cycles can be executed with a single trigger.

This register should not be written if previous host SPI write cycle is in progress. HSPI_MAIN_CS and HSPI_SUB_CS controls the main and sub display chip selects and therefore also determine how many SPI write cycles to main and sub displays and the write data position.

HSPI_MAIN_CS or HSPI_SUB_CS should be programmed to have at least one valid cycle when programming HSPI_CS_DC register.

Offset: 324h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0

Bit	Reset	Description
28:24	X	HSPI_RAISE_VECTOR: Host SPI Raise Vector This raise vector is returned after all the triggered host SPI cycles are executed.
19:16	X	HSPI_RAISE_CHANNEL_ID: Win G Channel ID
15:12	X	HSPI_SUB_CS: Host SPI Sub display Chip Select (Sub SCS_) 0= Sub display not selected (Sub SCS_=1) 1= Sub display selected (Sub SCS_=0) This is valid for Host SPI only. Each bit of this parameter corresponds to the four possible host SPI cycles.
11:8	X	HSPI_MAIN_CS: Host SPI Main display Chip Select (Main SCS_) 0= Main display not selected (Main SCS_=1) 1= Main display selected (Main SCS_=0) This is valid for Host SPI only. Each bit of this parameter corresponds to the four possible host SPI cycles.
7:4	X	HSPI_DC: Host SPI Data/Command_ (SDC) 0= Command cycle (SDC=0) 1= Data cycle (SDC=1) This is valid for Host SPI only. Each bit of this parameter corresponds to the four possible host SPI cycles.
0	0x0	HSPI_RAISE: Host SPI Raise. Raise vector will be returned at the end of the host SPI write cycles 0 = DISABLE 1 = ENABLE

29.8.38 DC_COM_SCRATCH_REGISTER_A_0

Scratch Register A

Class: Software Scratch Registers

Offset: 325h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	SCRATCH_REGISTER_A: Scratch Register A

29.8.39 DC_COM_SCRATCH_REGISTER_B_0

Scratch Register B

Offset: 326h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	SCRATCH_REGISTER_B: Scratch Register B

29.8.40 DC_COM_GPIO_CTRL_0

Display GPIO control, including debounce control

Offset: 327h | Read/Write: R/W | Reset: 0b000

Bit	Reset	Description
2	0x0	GPIO_2_DEBOUNCE_ENABLE: maps to display pin LCD_PWR2 0 = DISABLE 1 = ENABLE
1	0x0	GPIO_1_DEBOUNCE_ENABLE: maps to display pin LCD_PWR1 0 = DISABLE 1 = ENABLE
0	0x0	GPIO_0_DEBOUNCE_ENABLE: maps to display pin LCD_PWR0 0 = DISABLE 1 = ENABLE

29.8.41 DC_COM_GPIO_DEBOUNCE_COUNTER_0

Display GPIO Debounce Counter

Provides mark_en to debounce logic.

MARKER

The MARKER value sets the interval at which the debounce finite state machines associated with each GPIO input pin evaluate input transitions. An input transition must be stable for at least 4 consecutive MARKER ticks before it is propagated through the debounce circuit.

MARKER is in display clock tick units. The actual 'gravity delay' of a debounce FSM is 4 * Tmarker, where Tmarker = MARKER * (APB clock duration). For a 100 MHz display clock, a MARKER setting of 10,000,000 results in a gravity delay of 4x100 mS.

Offset: 328h | Read/Write: R/W | Reset: 0b00000000000000000000010000000000

Bit	Reset	Description
31:0	0x400	DEBOUNCE_COUNTER

29.8.42 DC_COM_CRC_CHECKSUM_LATCHED_0

CRC Checksum latched

This register is a latched version of CRC_CHECKSUM. Latching happens at frame end.

> **Note:** CRC_INPUT_DATA needs to be set to ACTIVE_DATA if this register is used. In full frame mode, CRC is frozen two cycles after frame end due to pipelining, so only in active area mode, CRC is consistent and independent of display control mode, and can be checked continuously frame by frame.

Offset: 329h | Read/Write: RO | Reset: 0x00000000

Bit	Reset	Description
31:0	0x0	CRC_CHECKSUM_LATCHED: CRC Checksum latched

29.9 Display DISP Registers

These registers control DISP display only; not including the DISP display window parameters.

29.9.1 DC_DISP_DISP_SIGNAL_OPTIONS0_0

Display Signal Options 0

Offset: 400h | Read/Write: R/W | Reset: 0b0x0xxx000x0xxx0x0x0

Bit	Reset	Description
26	0x0	M1_ENABLE: M1 Enable 0 = DISABLE 1 = ENABLE
24	0x0	M0_ENABLE: M0 Enable 0 = DISABLE 1 = ENABLE
20	0x0	V_PULSE3_ENABLE: V Pulse 3 Enable 0 = DISABLE 1 = ENABLE
19	0x0	V_PULSE2_ENABLE: V Pulse 2 Enable 0 = DISABLE 1 = ENABLE
18	0x0	V_PULSE1_ENABLE: V Pulse 1 Enable 0 = DISABLE 1 = ENABLE
16	0x0	V_PULSE0_ENABLE: V Pulse 0 Enable 0 = DISABLE 1 = ENABLE
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

Bit	Reset	Description
8	0x0	H_PULSE0_ENABLE: H Pulse 0 Enable 0 = DISABLE 1 = ENABLE

29.9.2 DC_DISP_DISP_SIGNAL_OPTIONS1_0

Display Signal Options 1

Offset: 401h | Read/Write: R/W | Reset: 0b0x0

Bit	Reset	Description
18	0x0	PP_ENABLE: PP Enable 0 = DISABLE 1 = ENABLE
16	0x0	DI_ENABLE: DI Enable 0 = DISABLE 1 = ENABLE

29.9.3 DC_DISP_DISP_WIN_OPTIONS_0

Display Window Options

Offset: 402h | Read/Write: R/W | Reset: 0b000xxxxxxxxxx0

Bit	Reset	Description
30	0x0	HDMI_ENABLE: HDMI interface. The HDMI unit must also be separately enabled in its own register space in order to use HDMI functionality. 0 = DISABLE 1 = ENABLE
29	0x0	DSI_ENABLE: MIPI Display Serial Interface Enable. The DSI unit must also be separately enabled in its own register space in order to use DSI functionality. 0 = DISABLE 1 = ENABLE
28	0x0	TVO_ENABLE: TVO Enable. Steps to start displaying on TV (The order of the first 3 steps can freely change): -- Program and enable TVO module -- Program DISPLAY_CTRL_MODE to NC_DISPLAY -- Program the ASSEMBLY shadow copy of this register field with ENABLE -- Program GENERAL_ACT_REQ to activate the shadow 0 = DISABLE 1 = ENABLE
16	0x0	CURSOR_ENABLE: Cursor Enable 0 = DISABLE 1 = ENABLE

29.9.4 DC_DISP_MEM_HIGH_PRIORITY_0

Memory High Priority request control

Display Memory High Priority Threshold

This controls high priority request for memory read access for each display window and for cursor. High priority request threshold should be increased in scenarios where memory access latency is high.

Offset: 403h | Read/Write: R/W | Reset: 0b000x0000000x000000000000000

Bit	Reset	Description
26:24	0x0	CSR_DISPLAYHC2MC_HPTH: Cursor Memory High Priority enable 0= memory access for cursor is normal priority 1= memory access for cursor is high priority
22:16	0x0	CBR_DISPLAY0C2MC_HPTH: Window C Memory High Priority threshold Memory access for this window is high priority if the number of filled entries in the return data fifo is less than or equal to this value. Setting this parameter to 0 disables high priority memory request.
14:8	0x0	CBR_DISPLAYB2MC_HPTH: Window B Memory High Priority threshold Memory access for this window is high priority if the number of filled entries in the return data fifo is less than or equal to this value. Setting this parameter to 0 disables high priority memory request. This register is used for both window B0 and B1
7:0	0x0	CBR_DISPLAY0A2MC_HPTH: Window A Memory High Priority threshold Memory access for this window is high priority if the number of filled entries in the return data fifo is less than or equal to this value. Setting this parameter to 0 disables high priority memory request.

29.9.5 DC_DISP_MEM_HIGH_PRIORITY_TIMER_0

Memory High Priority request control

Display Memory High Priority Timer

The high-priority assertion can be delayed by a number of memory clock cycles indicated by the timer. This creates an hysteresis effect, avoiding setting the high-priority for very short periods of time, which may or may not be desirable.

Offset: 404h | Read/Write: R/W | Reset: 0b000000xx000000xx000000xx000000

Bit	Reset	Description
29:24	0x0	CSR_DISPLAYHC2MC_HPTM: Cursor Memory High Priority timer
21:16	0x0	CBR_DISPLAY0C2MC_HPTM: Window C Memory High Priority timer
13:8	0x0	CBR_DISPLAYB2MC_HPTM: Window B Memory High Priority timer This register is used for both window B0 and B1
5:0	0x0	CBR_DISPLAY0A2MC_HPTM: Window A Memory High Priority timer

29.9.6 DC_DISP_DISP_TIMING_OPTIONS_0

Display Timing Options

Class: Display Standard Timings

Programming of display timing registers must meet these restrictions:

Constraint 1: H_REF_TO_SYNC + H_SYNC_WIDTH + H_BACK_PORCH > 11.

Constraint 2: V_REF_TO_SYNC + V_SYNC_WIDTH + V_BACK_PORCH > 1.

Constraint 3: V_FRONT_PORCH + V_SYNC_WIDTH + V_BACK_PORCH > 1 (vertical blank).

Constraint 4: V_SYNC_WIDTH >= 1

H_SYNC_WIDTH >= 1

Constraint 5: V_REF_TO_SYNC >= 1

H_REF_TO_SYNC >= 0

Constraint 6: V_FRONT_PORCH >= (V_REF_TO_SYNC + 1)

H_FRONT_PORCH >= (H_REF_TO_SYNC + 1)

Constraint 7: H_DISP_ACTIVE >= 16

V_DISP_ACTIVE >= 16

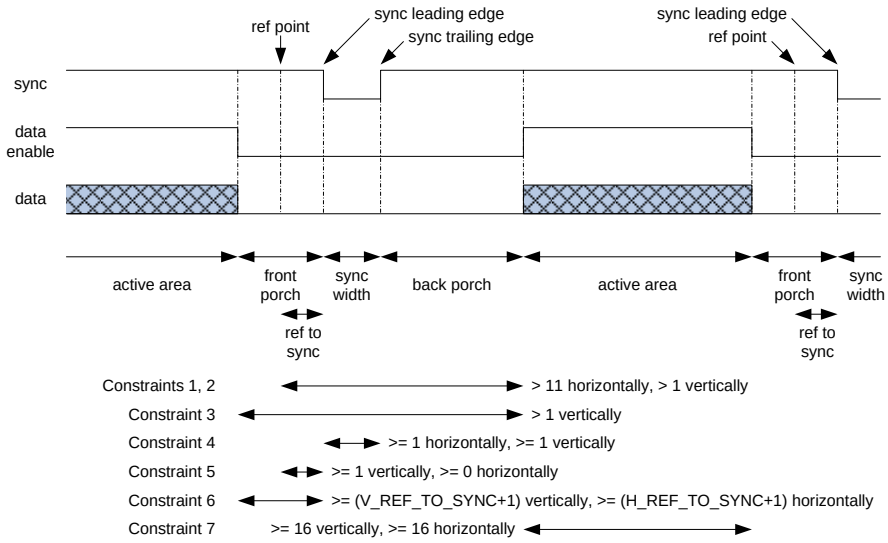
Only the back porch can be a negative value, though it is typically positive. Note that the back porch is defined as the distance between the trailing edge of the sync and the beginning of the active area. Some LCD specifications define the back porch from the leading edge of the sync rather than the trailing edge. Front porch is defined as the distance between the end of the active area and the leading edge of the sync. Vertical timing is in terms of lines and horizontal timing is in terms of pixels.

Sync polarity is in terms of the sync width; active low means the sync width pulse will be low, as in the diagram. Pixel clock polarity is in terms of when data will transition; an active low pixel clock means that the data changes on the falling edge of the clock, to be latched on the rising edge. Data enable (also called display enable or DE) polarity selects the level of the signal during the active area of display. The diagram has DE active high.

Timing Diagram

- This diagram applies to both vertical and horizontal timing
- Back porch is the only parameter that can be negative

Figure 92 Display Timing Options Timing Diagram



This register specifies display timing options for HSYNC and VSYNC

Offset: 405h | Read/Write: R/W | Reset: 0b0000000000000

Bit	Reset	Description
12:0	0x0	VSYNC_H_POSITION: VSYNC Horizontal Position This parameter specifies the position where VSYNC can toggle with respect to H reference point.

29.9.7 DC_DISP_REF_TO_SYNC_0

H/V Reference to Sync

This register specifies the start position of HSYNC and VSYNC with respect to H and V reference point (line and frame start) correspondingly. The H and V reference points correspond to the time when H and V display timing counter is re-initialized to zero correspondingly.

The H reference point also determines the point where V display timing counter is incremented so this points the horizontal relationship between HSYNC and VSYNC.

> **Note:** VSYNC's rising/falling edge is fixed at H reference point zero. In the future, we may want to add a horizontal position offset so that VSYNC can occur after HSYNC.

Offset: 406h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	V_REF_TO_SYNC: V reference to VSYNC (minimum 1 line clock)
12:0	X	H_REF_TO_SYNC: H reference to HSYNC (minimum 0 pixel clock)

29.9.8 DC_DISP_SYNC_WIDTH_0

H/V SYNC Pulse Width

This register specifies the width of HSYNC and VSYNC pulses. Check the comment for REF_TO_SYNC for programming restrictions.

Offset: 407h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	V_SYNC_WIDTH: VSYNC pulse width (minimum 1 line clock)
12:0	X	H_SYNC_WIDTH: HSYNC pulse width (minimum 1 pixel clock)

29.9.9 DC_DISP_BACK_PORCH_0

H/V Back Porch

This register specifies the distance between H/V SYNC trailing edge to beginning of display active area. This is 2's complement value and negative value indicates that H/V SYNC overlaps with the corresponding display active area. Check the comment for REF_TO_SYNC for programming restrictions.

Offset: 408h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	V_BACK_PORCH: V back porch
12:0	X	H_BACK_PORCH: H back porch

29.9.10 DC_DISP_DISP_ACTIVE_0

H/V Display Active width

This register specifies the width of H/V display active area. Check the comment for REF_TO_SYNC for programming restrictions.

Offset: 409h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	V_DISP_ACTIVE: V display active width (minimum 16 lines)
12:0	X	H_DISP_ACTIVE: H display active width (minimum 16 pixels)

29.9.11 DC_DISP_FRONT_PORCH_0

H/V Front Porch

This register specifies the distance between end of H/V display active area to the leading edge of the corresponding H/V SYNC.

Design Note: H/V active end plus the H/V front porch value minus the H/V reference to H/VSYNC determines the the H/V total (final H/V count value for the H/V display counter). Check the comment for REF_TO_SYNC for programming restrictions.

Offset: 40ah | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	V_FRONT_PORCH: VSYNC front porch (minimum -=PS_=-V_REF_TO_SYNC + 1)
12:0	X	H_FRONT_PORCH: HSYNC front porch (minimum -=PS_=-H_REF_TO_SYNC + 1)

29.9.12 DC_DISP_H_PULSE0_CONTROL_0

H Pulse 0 Control

Class: Display Extended Timings

Horizontal pulse 0 is programmable pulse that repeats every line.

In the NORMAL mode, this signal can have several pulses (A to D) per line with programmable width as defined by the pairs of start and end positions. The pulses must not overlap and must occur in sequence: pulse A, then pulse B, etc. In this case, the Enable field must be set to one of the End position. If the Enable field is set to one of the Start position then the pulse generator will stop as if the Enable field is set to the previous End position. If Enable field is set to Start A position then no pulse is generated.

In the ONE_CLOCK mode this signal can have up to twice the number of pulses per line with each pulse having a width of 1 pixel clock. In this mode, the position of the one-clock pulses correspond to the enabled Start and End positions.

Regardless of the mode, the pulse generator processes the pairs of start and end position sequentially in the order of: Start A, End A, Start B, End B, etc.

So these start/end positions should be programmed in increasing order. If any of the positions are programmed in non-increasing order (has invalid value) then the pulse generator will stop at the last valid position.

Polarity adjustment is made prior to V display qualification. This register specifies options for Horizontal pulse 0.

Offset: 40bh | Read/Write: R/W | Reset: 0bxxxxxxxxx

Bit	Reset	Description
11:8	X	H_PULSE0_LAST: H Pulse 0 Last point 0= end on Start A position 1= end on End A position 2= end on Start B position 3= end on End B position 4= end on Start C position 5= end on End C position 6= end on Start D position 7= end on End D position others= reserved 0 = START_A 1 = END_A 2 = START_B 3 = END_B 4 = START_C 5 = END_C 6 = START_D 7 = END_D
7:6	X	H_PULSE0_V_QUAL: H Pulse 0 Vertical Qualifier 0= always running 2= run during vertical active area 3= run during vertical active plus 1 line 0 = ALWAYS 2 = VACTIVE 3 = VACTIVE1
4	X	H_PULSE0_POLARITY: H Pulse 0 Polarity. Polarity adjustment is done before the vertical qualifier is applied. 0 = HIGH 1 = LOW
3	X	H_PULSE0_MODE: H Pulse 0 Mode 0= Normal mode 1= Single-clock mode 0 = NORMAL 1 = ONE_CLOCK

29.9.13 DC_DISP_H_PULSE0_POSITION_A_0

H Pulse 0 Position A

Offset: 40ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	H_PULSE0_END_A: H Pulse 0 End A (minimum -=PS_=-H_PULSE0_START_A+1)
12:0	X	H_PULSE0_START_A: H Pulse 0 Start A (minimum 0)

29.9.14 DC_DISP_H_PULSE0_POSITION_B_0

H Pulse 0 Position B

Offset: 40dh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	H_PULSE0_END_B: H Pulse 0 End B (minimum -=PS_=-H_PULSE0_START_B+1)
12:0	X	H_PULSE0_START_B: H Pulse 0 Start B (minimum -=PS_=-H_PULSE0_END_A+1)

29.9.15 DC_DISP_H_PULSE0_POSITION_C_0

H Pulse 0 Position C

Offset: 40eh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	H_PULSE0_END_C: H Pulse 0 End C (minimum -=PS_=-H_PULSE0_START_C+1)
12:0	X	H_PULSE0_START_C: H Pulse 0 Start C (minimum -=PS_=-H_PULSE0_END_B+1)

29.9.16 DC_DISP_H_PULSE0_POSITION_D_0

H Pulse 0 Position D

Offset: 40fh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	H_PULSE0_END_D: H Pulse 0 End D (minimum -=PS_=-H_PULSE0_START_D+1)
12:0	X	H_PULSE0_START_D: H Pulse 0 Start D (minimum -=PS_=-H_PULSE0_END_C+1)

29.9.17 DC_DISP_H_PULSE1_CONTROL_0

H Pulse 1 Control

Horizontal pulse 1 is programmable pulse that repeats every line.

In the NORMAL mode, this signal can have several pulses (A to D) per line with programmable width as defined by the pairs of start and end positions. The pulses must not overlap and must occur in sequence: pulse A, then pulse B, etc. In this case, the Enable field must be set to one of the End position. If the Enable field is set to one of the Start position then the pulse generator will stop as if the Enable field is set to the previous End position. If Enable field is set to Start A position then no pulse is generated.

In the ONE_CLOCK mode this signal can have up to twice the number of pulses per line with each pulse having a width of 1 pixel clock. In this mode, the position of the one-clock pulses correspond to the enabled Start and End positions.

Regardless of the mode, the pulse generator processes the pairs of start and end position sequentially in the order of: Start A, End A, Start B, End B, etc.

So these start/end positions should be programmed in increasing order. If any of the positions are programmed in non-increasing order (has invalid value) then the pulse generator will stop at the last valid position.

Polarity adjustment is made prior to V display qualification. This register specifies options for Horizontal pulse 1.

Offset: 410h | Read/Write: R/W | Reset: 0bxxxxxxxxx

Bit	Reset	Description
11:8	X	H_PULSE1_LAST: H Pulse 1 Last point 0= end on Start A position 1= end on End A position 2= end on Start B position 3= end on End B position 4= end on Start C position 5= end on End C position 6= end on Start D position 7= end on End D position others= reserved 0 = START_A 1 = END_A 2 = START_B 3 = END_B 4 = START_C 5 = END_C 6 = START_D 7 = END_D
7:6	X	H_PULSE1_V_QUAL: H Pulse 1 Vertical Qualifier 0= always running 2= run during vertical active area 3= run during vertical active plus 1 line 0 = ALWAYS 2 = VACTIVE 3 = VACTIVE1
4	X	H_PULSE1_POLARITY: H Pulse 1 Polarity. Polarity adjustment is done before the vertical qualifier is applied 0 = HIGH 1 = LOW
3	X	H_PULSE1_MODE: H Pulse 1 Mode 0= Normal mode 1= Single-clock mode 0 = NORMAL 1 = ONE_CLOCK

29.9.18 DC_DISP_H_PULSE1_POSITION_A_0

H Pulse 1 Position A

Offset: 411h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	H_PULSE1_END_A: H Pulse 1 End A (minimum -=PS_=-H_PULSE1_START_A+1)
12:0	X	H_PULSE1_START_A: H Pulse 1 Start A (minimum 0)

29.9.19 DC_DISP_H_PULSE1_POSITION_B_0

H Pulse 1 Position B

Offset: 412h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	H_PULSE1_END_B: H Pulse 1 End B (minimum -=PS_=-H_PULSE1_START_B+1)
12:0	X	H_PULSE1_START_B: H Pulse 1 Start B (minimum -=PS_=-H_PULSE1_END_A+1)

29.9.20 DC_DISP_H_PULSE1_POSITION_C_0

H Pulse 1 Position C

Offset: 413h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	H_PULSE1_END_C: H Pulse 1 End C (minimum -=PS_=-H_PULSE1_START_C+1)
12:0	X	H_PULSE1_START_C: H Pulse 1 Start C (minimum -=PS_=-H_PULSE1_END_B+1)

29.9.21 DC_DISP_H_PULSE1_POSITION_D_0

H Pulse 1 Position D

Offset: 414h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	H_PULSE1_END_D: H Pulse 1 End D (minimum -=PS_=-H_PULSE1_START_D+1)
12:0	X	H_PULSE1_START_D: H Pulse 1 Start D (minimum -=PS_=-H_PULSE1_END_C+1)

29.9.22 DC_DISP_H_PULSE2_CONTROL_0

H Pulse 2 Control

Horizontal pulse 2 is programmable pulse that repeats every line.

In the NORMAL mode, this signal can have several pulses (A to D) per line with programmable width as defined by the pairs of start and end positions. The pulses must not overlap and must occur in sequence: pulse A, then pulse B, etc. In this case, the Enable field must be set to one of the End position. If the Enable field is set to one of the Start position then the pulse generator will stop as if the Enable field is set to the previous End position. If Enable field is set to Start A position then no pulse is generated.

In the ONE_CLOCK mode this signal can have up to twice the number of pulses per line with each pulse having a width of 1 pixel clock. In this mode, the position of the one-clock pulses correspond to the enabled Start and End positions.

Regardless of the mode, the pulse generator processes the pairs of start and end position sequentially in the order of: Start A, End A, Start B, End B, etc.

So these start/end positions should be programmed in increasing order. If any of the positions are programmed in non-increasing order (has invalid value) then the pulse generator will stop at the last valid position.

Polarity adjustment is made prior to V display qualification. This register specifies options for Horizontal pulse 2.

Offset: 415h | Read/Write: R/W | Reset: 0bxxxxxxxxxx

Bit	Reset	Description

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

29.9.23 DC_DISP_H_PULSE2_POSITION_A_0

H Pulse 2 Position A

Offset: 416h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	H_PULSE2_END_A: H Pulse 2 End A (minimum -=PS_=-H_PULSE2_START_A+1)
12:0	X	H_PULSE2_START_A: H Pulse 2 Start A (minimum 0)

29.9.24 DC_DISP_H_PULSE2_POSITION_B_0

H Pulse 2 position B

Offset: 417h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	H_PULSE2_END_B: H Pulse 2 End B (minimum -=PS_=-H_PULSE2_START_B+1)
12:0	X	H_PULSE2_START_B: H Pulse 2 Start B (minimum -=PS_=-H_PULSE2_END_A+1)

29.9.25 DC_DISP_H_PULSE2_POSITION_C_0

H Pulse 2 Position C

Offset: 418h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	H_PULSE2_END_C: H Pulse 2 End C (minimum -=PS_=-H_PULSE2_START_C+1)
12:0	X	H_PULSE2_START_C: H Pulse 2 Start C (minimum -=PS_=-H_PULSE2_END_B+1)

29.9.26 DC_DISP_H_PULSE2_POSITION_D_0

H Pulse 2 Position D

Offset: 419h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	H_PULSE2_END_D: H Pulse 2 End D (minimum -=PS_=-H_PULSE2_START_D+1)
12:0	X	H_PULSE2_START_D: H Pulse 2 Start D (minimum -=PS_=-H_PULSE2_END_C+1)

29.9.27 DC_DISP_V_PULSE0_CONTROL_0

V Pulse 0 Control

Vertical pulse 0 is programmable pulse that repeats every frame.

This signal can have several pulses (A to C) per frame with programmable width as defined by the pairs of start and end positions. The pulses must not overlap and must occur in sequence: pulse A, then pulse B, etc. In this case, the Enable field must be set to one of the End position. If the Enable field is set to one of the Start position then the pulse generator will stop as if the Enable field is set to the previous End position. If Enable field is set to Start A position then no pulse is generated.

The pulse generator processes the pairs of start and end position sequentially in the order of: Start A, End A, Start B, End B, etc. So these start/end positions should be programmed in increasing order. If any of the positions are programmed in non-increasing order (has invalid value) then the pulse generator will stop at the last valid position.

This register specifies options for Vertical pulse 0.

Offset: 41ah | Read/Write: R/W | Reset: 0b0000000000000xxxxxxxxxxxx

Bit	Reset	Description
28:16	0x0	V_PULSE0_H_POSITION: V Pulse 0 Horizontal Position This parameter specifies the position where V Pulse 0 can toggle with respect to H reference point.
11:8	X	V_PULSE0_LAST: V Pulse 0 Last point 0= end on Start A position 1= end on End A position 2= end on Start B position 3= end on End B position 4= end on Start C position 5= end on End C position others= reserved 0 = START_A 1 = END_A 2 = START_B 3 = END_B 4 = START_C 5 = END_C

Bit	Reset	Description
7:6	X	V_PULSE0_DELAY: V Pulse 0 Delay 0= no delay 1= 1-line delay 2= 2-line delay 3= reserved 0 = NODELAY 1 = DELAY1 2 = DELAY2
4	X	V_PULSE0_POLARITY: V Pulse 0 Polarity 0= High 1= Low 0 = HIGH 1 = LOW

29.9.28 DC_DISP_V_PULSE0_POSITION_A_0

V Pulse 0 Position A

Offset: 41bh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	V_PULSE0_END_A: V Pulse 0 End A (minimum -=PS_=-V_PULSE0_START_A+1)
12:0	X	V_PULSE0_START_A: V Pulse 0 Start A (minimum 0)

29.9.29 DC_DISP_V_PULSE0_POSITION_B_0

V Pulse 0 Position B

Offset: 41ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	V_PULSE0_END_B: V Pulse 0 End B (minimum -=PS_=-V_PULSE0_START_B+1)
12:0	X	V_PULSE0_START_B: V Pulse 0 Start B (minimum -=PS_=-V_PULSE0_END_A+1)

29.9.30 DC_DISP_V_PULSE0_POSITION_C_0

V Pulse 0 Position C

Offset: 41dh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	V_PULSE0_END_C: V Pulse 0 End C (minimum -=PS_=-V_PULSE0_START_C+1)
12:0	X	V_PULSE0_START_C: V Pulse 0 Start C (minimum -=PS_=-V_PULSE0_END_B+1)

29.9.31 DC_DISP_V_PULSE1_CONTROL_0

V pulse 1 Control

Vertical pulse 1 is programmable pulse that repeats every frame. This signal can have several pulses (A to C) per frame with programmable width as defined by the pairs of start and end positions. The pulses must not overlap and must occur in sequence: pulse A, then pulse B, etc.

In this case, the Enable field must be set to one of the End position. If the Enable field is set to one of the Start position then the pulse generator will stop as if the Enable field is set to the previous End position. If Enable field is set to Start A position then no pulse is generated.

The pulse generator processes the pairs of start and end position sequentially in the order of: Start A, End A, Start B, End B, etc.

So these start/end positions should be programmed in increasing order. If any of the positions are programmed in non-increasing order (has invalid value) then the pulse generator will stop at the last valid position.

This register specifies options for Vertical pulse 1.

Offset: 41eh | Read/Write: R/W | Reset: 0b0000000000000xxxxxxxxxxxx

Bit	Reset	Description
28:16	0x0	V_PULSE1_H_POSITION: V Pulse 1 Horizontal Position This parameter specifies the position where V Pulse 1 can toggle with respect to H reference point.
11:8	X	V_PULSE1_LAST: V pulse 1 Last point 0= end on Start A position 1= end on End A position 2= end on Start B position 3= end on End B position 4= end on Start C position 5= end on End C position others= reserved 0 = START_A 1 = END_A 2 = START_B 3 = END_B 4 = START_C 5 = END_C
7:6	X	V_PULSE1_DELAY: V pulse 1 Delay 0= no delay 1= 1-line delay 2= 2-line delay 3= reserved 0 = NODELAY 1 = DELAY1 2 = DELAY2
4	X	V_PULSE1_POLARITY: V pulse 1 Polarity 0 = HIGH 1 = LOW

29.9.32 DC_DISP_V_PULSE1_POSITION_A_0

V Pulse 1 Position A

Offset: 41fh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	V_PULSE1_END_A: V Pulse 1 End A (minimum -=PS_=-V_PULSE1_START_A+1)
12:0	X	V_PULSE1_START_A: V Pulse 1 Start A (minimum 0)

29.9.33 DC_DISP_V_PULSE1_POSITION_B_0

V Pulse 1 Position B

Offset: 420h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	V_PULSE1_END_B: V Pulse 1 End B (minimum -=PS_=-V_PULSE1_START_B+1)
12:0	X	V_PULSE1_START_B: V Pulse 1 Start B (minimum -=PS_=-V_PULSE1_END_A+1)

29.9.34 DC_DISP_V_PULSE1_POSITION_C_0

V Pulse 1 Position C

Offset: 421h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	V_PULSE1_END_C: V Pulse 1 End C (minimum -=PS_=-V_PULSE1_START_C+1)
12:0	X	V_PULSE1_START_C: V Pulse 1 Start C (minimum -=PS_=-V_PULSE1_END_B+1)

29.9.35 DC_DISP_V_PULSE2_CONTROL_0

V pulse 2 Control

Vertical pulse 2 is programmable pulse that repeats every frame.

This signal can have one pulse (A) per frame with programmable width as defined by the pair of start and end positions.

In this case, the Enable field must be set to one of the End position. If the Enable field is set to one of the Start position then the pulse generator will stop as if the Enable field is set to the previous End position. If Enable field is set to Start A position then no pulse is generated.

So these start/end positions should be programmed in increasing order. If any of the positions are programmed in non-increasing order (has invalid value) then the pulse generator will stop at the last valid position.

This register specifies options for Vertical pulse 2.

Offset: 422h | Read/Write: R/W | Reset: 0b0000000000000xxxxxxxxxxxx

Bit	Reset	Description
28:16	0x0	V_PULSE2_H_POSITION: V Pulse 2 Horizontal Position This parameter specifies the position where V Pulse 2 can toggle with respect to H reference point.
8	X	V_PULSE2_LAST: V pulse 2 Last point 0= end on Start A position 1= end on End A position others= reserved 0 = START_A 1 = END_A
4	X	V_PULSE2_POLARITY: V pulse 2 Polarity 0 = HIGH 1 = LOW

29.9.36 DC_DISP_V_PULSE2_POSITION_A_0

V Pulse 2 Position A

Offset: 423h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	V_PULSE2_END_A: V Pulse 2 End A (minimum -=PS_=-V_PULSE2_START_A+1)
12:0	X	V_PULSE2_START_A: V Pulse 2 Start A (minimum 0)

29.9.37 DC_DISP_V_PULSE3_CONTROL_0

V pulse 3 Control

Vertical pulse 3 is programmable pulse that repeats every frame.

This signal can have one pulse (A) per frame with programmable width as defined by the pair of start and end positions.

In this case, the Enable field must be set to one of the End position. If the Enable field is set to one of the Start position then the pulse generator will stop as if the Enable field is set to the previous End position. If Enable field is set to Start A position then no pulse is generated.

So these start/end positions should be programmed in increasing order. If any of the positions are programmed in non-increasing order (has invalid value) then the pulse generator will stop at the last valid position.

This register specifies options for Vertical pulse 2.

Offset: 424h | Read/Write: R/W | Reset: 0b0000000000000xxxxxxxxxxx0

Bit	Reset	Description
28:16	0x0	V_PULSE3_H_POSITION: V Pulse 3 Horizontal Position This parameter specifies the position where V Pulse 3 can toggle with respect to H reference point.
8	X	V_PULSE3_LAST: V pulse 3 Last point 0= end on Start A position 1= end on End A position others= reserved 0 = START_A 1 = END_A
4	0x0	V_PULSE3_POLARITY: V pulse 3 Polarity 0 = HIGH 1 = LOW

29.9.38 DC_DISP_V_PULSE3_POSITION_A_0

V Pulse 3 Position A

Offset: 425h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	V_PULSE3_END_A: V Pulse 3 End A (minimum -=PS_=-V_PULSE3_START_A+1)
12:0	X	V_PULSE3_START_A: V Pulse 3 Start A (minimum 0)

29.9.39 DC_DISP_M0_CONTROL_0

M0 Control

Display M0 signal

M0 signal can be generated either using a line (horizontal) or a frame (vertical) clock and it can be horizontally positioned with respect to H reference point. This signal is typically output on LCD_CS1_N pin.

This register specifies options for M0 signal.

Offset: 426h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description

Bit	Reset	Description
28:16	X	M0_H_POSITION: M0 Horizontal Position This parameter specifies the position where M0 can toggle with respect to H reference point.
12:8	X	M0_PERIOD: M0 Period This should be program to the half of the desired M0 period (in lines) minus 1.
7	X	M0_POLARITY: M0 Polarity. Polarity adjustment is applied last after phase control is applied. 0 = HIGH 1 = LOW
6	X	M0_PHASE_RESET: M0 Phase Reset This bit is effective only when M0 is not free running. 0= frequency (phase) counter is not reset 1= frequency (phase) counter is reset at beginning of vertical active display if phase control is set to VACTIVE_RESTART or at beginning of frame if phase control is set to FRAME_INVERT 0 = DISABLE 1 = ENABLE
5:4	X	M0_PHASE_CONTROL: M0 Phase Control 00= free-running 01= reserved 10= reset at beginning of vertical active display 11= invert at beginning of frame This should be set to free-running if frame clock is used. 0 = FREE_RUN 2 = VACTIVE_RESTART 3 = FRAME_INVERT
1:0	X	M0_CLOCK_SELECT: M0 Clock Select 00= pixel clock (for diagnostic) 01= reserved 10= line clock 11= frame clock 0 = PCLK 2 = LCLK 3 = FCLK

29.9.40 DC_DISP_M1_CONTROL_0

M1 Control

Display M1 signal. M1 signal can be generated either using a line (horizontal) or a frame (vertical) clock and it can be horizontally positioned with respect to H reference point. This signal is typically output on LCD_M1 pin. This register specifies options for M1 signal.

Offset: 427h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	M1_H_POSITION: M1 Horizontal Position This parameter specifies the position where M0 can toggle with respect to H reference point.
12:8	X	M1_PERIOD: M1 Period This should be program to the half of the desired M1 period (in lines) minus 1.
7	X	M1_POLARITY: M1 Polarity. Polarity adjustment is applied last after phase control is applied. 0 = HIGH 1 = LOW
6	X	M1_PHASE_RESET: M1 Phase Reset This bit is effective only when M1 is not free running. 0= frequency (phase) counter is not reset 1= frequency (phase) counter is reset at beginning of vertical active display if phase control is set to VACTIVE_RESTART or at beginning of frame if phase control is set to FRAME_INVERT 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
5:4	X	M1_PHASE_CONTROL: M1 Phase Control 00= free-running 01= reserved 10= reset at beginning of vertical active display 11= invert at beginning of frame This should be set to free-running if frame clock is used. 0 = FREE_RUN 2 = VACTIVE_RESTART 3 = FRAME_INVERT
1:0	X	M1_CLOCK_SELECT: M1 Clock Select 00= pixel clock (for diagnostic) 01= synchronous to M0 provided that M0 is generated using line clock. This will not work if M0 is not generated using line clock. In this case, M1 is controlled by -=PS_=-M0_PHASE_RESET and -=PS_=-M0_PERIOD, -=PS_=-M1_PHASE_CONTROL and -=PS_=-M1_POLARITY. 10= line clock 11= frame clock 0 = PCLK 1 = M0SYNC 2 = LCLK 3 = FCLK

29.9.41 DC_DISP_DI_CONTROL_0

DI Control

Display Data Inversion (DI) signal generation.

This signal is typically needed to control data inversion for PWM panels and is typically output on LCD_D22 pin.

Horizontal position of this signal with respect to horizontal reference point can be programmed. DI signal together with M0 may also be used to control the actual pixel data inversion.

Pixel data may be controlled by either DI only or by (DI ^ M0) as specified by -=PS_=-PIXDATA_INV_SELECT. The inversion control signal is then used to control pixel data inversion as specified by -=PS_=-PIXDATA_INV_CONTROL. Note that even if the DI signal is disabled, pixel data inversion could still occur depending on the setting of -=PS_=-PIXDATA_INV_CONTROL.

Data inversion is limited to only active area. For the purpose of pixel data inversion, DI and M0 signals are used before the corresponding horizontal positioning so that these signals are always stable during active area.

In case M0 signal is used to control data inversion then it should be generated using line clock. M0 polarity control is not accounted when M0 is used to generate DI signal or to control pixel data inversion.

This register specifies options for DI signal as well as pixel data inversion.

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

Bit	Reset	Description
28:16	X	DI_H_POSITION: DI signal Horizontal Position This parameter specifies the position where DI signal can toggle with respect to H reference point. It should not be programmed larger than -=PS_=-PP_H_POSITION if DI is used to control PP signal generation.
7:6	X	PIXDATA_INV_CONTROL: Pixel Data Inversion Control The control signal for pixel data inversion is defined by -=PS_=-PIXDATA_INV_SELECT 00= no pixel data inversion regardless of control signal state. 01= Pixels 0, 2, 4 ... are inverted if control signal is high. Pixels 1, 3, 5 ... are inverted if control signal is low. 10= Pixels 1, 3, 5 ... are inverted if control signal is high. Pixels 0, 2, 4 ... are inverted if control signal is low. 11= all pixel data is inverted if control signal is high. NOTE: Pixel data inversion is NOT supported for 2-pixel/3-clock 12-bit parallel display data format!!! 0 = NOINV 1 = EVENINV 2 = ODDINV 3 = ALLINV

Bit	Reset	Description
4	X	PIXDATA_INV_SELECT: Pixel Data Inversion Select 0= DI signal controls pixel data inversion 1= (DI xor M0) controls pixel data inversion. 0 = DI 1 = DIXORM0
1:0	X	DI_MODE: DI signal Mode 00= DI is always low 01= DI is always high 10= DI is forced high every time M0 (before polarity adjustment) toggles from low to high; otherwise then DI toggles every line 11= DI has same frequency (phase) as M0 (before M0 polarity adjustment)

29.9.42 DC_DISP_PP_CONTROL_0

PP Control

Display Programmable Pulse (PP) signal generation

PP signal generation logic can generate up to 128 pulses per line internally and the PP pulse select registers determines which of the 128 pulses will be output. Any of the 128 internally generated pulse can be independently selected as output if they occur within one line time.

PP signal is typically output on LCD_D23 pin. Note that DI signal may impact PP generation as controlled by -=PS_=-PP_REVERSAL_CONTROL.

PP signal generation may be delayed (positioned) from H reference point (line start) controlled by -=PS_=-PP_H_DELAY. Delaying PP may cause the last few internal PP pulses to overflow to the next line.

PP is always generated using the display clock after the shift clock divider.

This register specifies options for PP signal.

Offset: 429h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:12	X	PP_LOW_PULSE: PP Low Pulse width (1 to 16)
11:8	X	PP_HIGH_PULSE: PP High Pulse width (1 to 16)
7:4	X	PP_H_DELAY: PP signal Horizontal Delay (0 to 15) This parameter specifies the position where PP signal generation starts with respect to H reference point. If DI is used to generate PP signal then this parameter should not be smaller than -=PS_=-DI_H_POSITION.
3:2	X	PP_V_QUALIFIER: PP Vertical Qualifier 0= free running (not qualified) 1= V Pulse 1 qualified 2= V Pulse 2 qualified 3= V Pulse 3 qualified 0 = FREE_RUN 1 = VPULSE1 2 = VPULSE2 3 = VPULSE3
1:0	X	PP_DIRECTION: PP Direction (incrementing or decrementing) 0= always from pulse 0 to 127 (regardless of DI signal) 1= 0 to 127 if DI=0 and 127 to 0 if DI=1 2= 127 to 0 if DI=0 and 0 to 127 if DI=1 3= always 127 to 0 regardless of DI 0 = ALWAYS_INC 1 = INC_IF_DI0 2 = DEC_IF_DI0 3 = ALWAYS_DEC

29.9.43 DC_DISP_PP_SELECT_A_0

PP Select A

The next 4 registers and -=PS_=-PP_DIRECTION which of the internal 128 pulses to be output.

Each bit in the four registers corresponds to one internal pulse.

Offset: 42ah | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	PP_SELECT_A: PP Select bits 31 to 0

29.9.44 DC_DISP_PP_SELECT_B_0

PP Select B

Offset: 42bh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	PP_SELECT_B: PP Select bits 63 to 32

29.9.45 DC_DISP_PP_SELECT_C_0

PP Select C

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

Bit	Reset	Description
31:0	X	PP_SELECT_C: PP Select bits 95 to 64

29.9.46 DC_DISP_PP_SELECT_D_0

PP Select D

Offset: 42dh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	PP_SELECT_D: PP Select bits 127 to 96

29.9.47 DC_DISP_DISP_CLOCK_CONTROL_0

Display Clock Control

Shift clock divider is used to divide root clock for display controller module to generate internal shift clock for shifting data to the display. Output of this divider is typically used to generate the external shift clock which is sent to the display (SC0 and/or SC1) except for 1-pixel/1-clock parallel display.

The output of this divider is also used to generate Programmable Pulse (PP) signal. For 1-pixel/1-clock parallel display, SC0 and SC1 are generated using the output of pixel clock divider which can be set to 1, 2, or 4 for 1-pixel/1-clock parallel display.

The reason pixel clock divider 2 and 4 are allowed for 1-pixel/1-clock parallel display interface is so that the clock that generates PP can be generated with 2x or 4x higher frequency than pixel clock and therefore can produce higher resolution PP pulse positions. For all cases of parallel display, SC0 and SC1 can be further divided by 1, 2 or 4.

Class: Display Interface Settings

This register controls generation of shift clock to the display and internal pixel clock. Internal display pipeline runs with pixel clock and processes 1 pixel per clock.

Offset: 42eh | Read/Write: R/W | Reset: 0b000000000110

Bit	Reset	Description
11:8	0x0	PIXEL_CLK_DIVIDER: Pixel Clock Divider 0000= divide by 1 0001= divide by 1.5 0010= divide by 2 0011= divide by 3 0100= divide by 4 0101= divide by 6 0110= divide by 8 0111= divide by 9 1000= divide by 12 1001= divide by 16 1010= divide by 18 1011= divide by 24 1100= divide by 13 other= reserved 0 = PCD1 1 = PCD1H 2 = PCD2 3 = PCD3 4 = PCD4 5 = PCD6 6 = PCD8 7 = PCD9 8 = PCD12 9 = PCD16 10 = PCD18 11 = PCD24 12 = PCD13
7:0	0x6	SHIFT_CLK_DIVIDER: Shift Clock Divider 0 = divide by 1 1 = divide by 1.5 2 = divide by 2 3 = divide by 2.5 4 = divide by 3 : : : : 254 = divide by 128 255 = divide by 128.5 Pixel clock divider is used to divide output of internal shift clock divider to generate internal pixel clock which is used to clock the internal horizontal and vertical counters. This divider also determines the output format for parallel interface, serial interface, and LCD SPI interface in conjunction with Display Data Format parameter. For 1-pixel/1-clock parallel display interface, valid settings are PCD1, PCD2, and PCD4. Note that the main reason to use PCD2 and PCD4 is to get higher frequency PP clock because the PP clock is always generated from the output of shift clock divider. For non 1-pixel/1-clock parallel display interface, valid settings are, PCD1H (2-pixel/3-clock), PCD2 (1-pixel/2-clock), and PCD3 (1-pixel/3-clock). For 1-channel serial display interface, valid settings are PCD3 (3-bpp 1-ch), PCD4 (3-bpp 1-ch), PCD6 (6-bpp 1-ch), PCD9 (9-bpp 1-ch), PCD12 (12-bpp 1-ch), PCD16 (16-bpp 1-ch), PCD18 (18-bpp 1-ch). For 2-channel serial display interface, valid settings are PCD2 (3-bpp 2-ch), PCD3 (6-bpp 2-ch), PCD6 (12-bpp 2-ch), PCD8 (16-bpp 2-ch), PCD9 (18-bpp 2-ch). For 3-channel serial display interface, valid settings are PCD1 (3-bpp 3-ch), PCD2 (6-bpp 3-ch), PCD3 (9-bpp 3-ch), PCD4 (12-bpp 3-ch), PCD6 (18-bpp 3-ch). For LCD SPI interface, valid settings are PCD12 (B4G4R4), PCD16 (B5G6R5), PCD18 (B6G6R6), PCD24 (B8G8R8), PCD8 (B5G6R5 with data/command bit), PCD6 (B5G6R5 with data/command start byte - depending on data/command bit), PCD4 (P8 for spi8), PCD9 (B5G6R5 with chip select deassertion at 8-bit boundary, spi16x2), PCD3 (P8 for spidc), PCD2 (B5G6R5 with data/command bit and chip select deassertion at 9-bit boundary, spi16x2dc), and PCD13 (spi12p2, no chip select deassertion between pairs of pixels).

29.9.48 DC_DISP_DISP_INTERFACE_CONTROL_0

Display Interface Control

This register specifies display interface options

Offset: 42fh | Read/Write: R/W | Reset: 0b00xxxx0000

Bit	Reset	Description
9	0x0	DISP_DATA_ORDER: Display Data Order This is effective only for 1-pixel/2-clock 16-/18-/24- bit parallel interface 0= Red pixel is output in the first clock and blue pixel is output in the second cycle 1= Blue pixel is output in the first clock cycle and red pixel is output in the second clock cycle 0 = RED_BLUE 1 = BLUE_RED
8	0x0	DISP_DATA_ALIGNMENT: Display Data Alignment This is effective for parallel display data format and the associated Initialization Sequence (IS). 0= Output data is MSB-aligned For 1-pixel/1-clock parallel display the output data ordering is the same regardless of display Base Color Size. For 1-pixel/1-clock parallel display data alignment is optimized for 18-bpp so the 24-bit data ordering is: LD[5:0] is blue data bits 7-2 LD[11:6] is green data bits 7-2 LD[17:12] is red data bits 7-2 LD[19:18] is blue data bits 1-0 LD[21:20] is green data bits 1-0 LD[23:22] is red data bits 1-0 Note that LD18 to LD23 signals are multiplexed with control pins (see Pin Output Select definition) 1= Output data is LSB-aligned. For 1-pixel/1-clock parallel display the output data ordering is determined by display Base Color Size. For 1-pixel/1-clock parallel display data alignment is optimized for 24-bpp as follows: LD[7:0] is blue data bits 7-0 LD[15:8] is green data bits 7-0 LD[23:16] is red data bits 7-0 Note that LD18 to LD23 signals are multiplexed with control pins (see Pin Output Select definition) 0 = MSB 1 = LSB
3:0	0x0	DISP_DATA_FORMAT: Display Data Format Pixel Clock Divider is used together with this parameter to determine the exact display data format. 0 = DF1P1C : 0= 1-pixel/1-clock up to 24-bit parallel 1 = DF1P2C24B : 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

29.9.49 DC_DISP_DISP_COLOR_CONTROL_0

Display Color Control

Offset: 430h | Read/Write: R/W | Reset: 0b0000xxxxx0x0xx00xxxxxxxx0000

Bit	Reset	Description
27	0x0	LCD_MD3: LCD Mode 3 signal 0 = LOW 1 = HIGH
26	0x0	LCD_MD2: LCD Mode 2 signal 0 = LOW 1 = HIGH
25	0x0	LCD_MD1: LCD Mode 1 signal 0 = LOW 1 = HIGH
24	0x0	LCD_MD0: LCD Mode 0 signal 0 = LOW 1 = HIGH

Bit	Reset	Description
18	0x0	NON_BASE_COLOR: Non Base Color 0= zeros 1= ones MD0-3 signals are general purpose mode signals that can be output in various pins (see Pin Output Select) to configure the display device. These bits are effective at start of frame. Typically these can be programmed in shadow register which takes effect on the next frame.
17	X	BLANK_COLOR: Blank Color 0= zeros 1= ones Non Base Color applies to least significant color bits which are not part of base color and it has higher priority over Border Color but lower priority over Blank color.
16	0x0	DISP_COLOR_SWAP: Display Color Swap 0= RGB (normal) 1= BGR (red-blue reverse) 0 = RGB 1 = BGR
13:12	0x0	ORD_DITHER_ROTATION: Ordered Dither Frame Rotation This parameter specifies the rotation frequency of the dither matrix in terms of number of frames. If programmed to 0, there is no dither matrix rotation. If programmed to N where N is larger than 0, the dither matrix is rotated clockwise every N frame.
9:8	X	DITHER_CONTROL: Dither Control 00= dither disabled 01= reserved 10= ordered dither 11= error-diffusion dither Design Note: initial dither matrix (where d is 2 dither bits) d=00 d=01 d=10 d=11 --------- --------- --------- --------- 0 0 1 0 0 1 0 1 --------- --------- --------- --------- 0 0 0 0 1 0 1 1 --------- --------- --------- --------- Note: 0 in the matrix specifies no addition to base color 1 in the matrix specifies incrementation of base color (with saturation) 0 = DISABLE 2 = ORDERED 3 = ERRDIFF
3:0	0x0	BASE_COLOR_SIZE: Display Base Color Size This parameter determines the number of bits per color after dither. 0= 6 bits 1= 1 bit 2= 2 bits 3= 3 bits 4= 4 bits 5= 5 bits 6= 5 bits for R,B and 6 bits for G 7= 3 bits for R,G and 2 bits for B 8= 8 bits, this also forces dither to be disabled. This setting can be used to output 24-bit data in 1-pixel/clock parallel display data format. 0 = BASE666 1 = BASE111 2 = BASE222 3 = BASE333 4 = BASE444 5 = BASE555 6 = BASE565 7 = BASE332 8 = BASE888

29.9.50 DC_DISP_SHIFT_CLOCK_OPTIONS_0

Shift Clock options

This register specifies options for both display shift clock 0 (SC0) and display shift clock 1 (SC1). SC0 signal is typically output on LCD_PCLK pin and SC1 signal is typically output on LCD_WR_N pin.

Offset: 431h | Read/Write: R/W | Reset: 0b00000000xxxxxxxx00000000

Bit	Reset	Description
23:22	0x0	SC1_CLK_DIVIDER: SC1 Clock Divider 0= divide by 1 - this is valid for all display interface 1= divide by 2 - this is valid only for 1-pixel/1-clock parallel display and 2-pixel/1-clock parallel display 2= divide by 4 - this is valid only for 1-pixel/1-clock parallel display 3= reserved 0 = DIV1 1 = DIV2 2 = DIV4

Bit	Reset	Description
21:19	0x0	SC1_V_QUALIFIER: SC1 Vertical Qualifier 0= no vertical qualifier 2= vertical display active 3= 1-line extended vertical display active 4= V Pulse 1 (VP1) 5= 1-line extended V Pulse 1 others= reserved If SC1 is divided by 2 then it is synchronously reset at the beginning of the horizontal qualifier such that rising edge of SC1 is generated for the first horizontally qualified 'pixel'. If SC1 is divided by 4 then it is synchronously reset at the beginning of the horizontal qualifier such that rising edge of LCD_WR_N is generated for the second horizontally qualified 'pixel'. In the case where there is no horizontal qualifier start of horizontal display active will be used to generate the synchronous reset. If Initialization Sequence (IS) is enabled on parallel interface then only divide by 1 is allowed for SC1 Clock Divider and SC1 must have vertical and horizontal qualifiers enabled. 0 = NO_VQUAL 1 = RESERVED 2 = VACTIVE 3 = EXT_VACTIVE 4 = VPULSE1 5 = EXT_VPULSE1
18:16	0x0	SC1_H_QUALIFIER: SC1 Horizontal Qualifier 0= disable (regardless of vertical qualifier) 1= no horizontal qualifier (V qualifier only) 2= horizontal display active 3= 1-clock early & extended H display active 4= H Pulse 1 (HP1) 5= 1-clock early & extended H Pulse 1 others= reserved 0 = DISABLE 1 = NO_HQUAL 2 = HACTIVE 3 = EXT_HACTIVE 4 = HPULSE1 5 = EXT_HPULSE1
7:6	0x0	SC0_CLK_DIVIDER: SC0 Clock Divider 0= divide by 1 - this is valid for all display interface 1= divide by 2 - this is valid only for 1-pixel/1-clock parallel display and 2-pixel/1-clock parallel display 2= divide by 4 - this is valid only for 1-pixel/1-clock parallel display 3= reserved 0 = DIV1 1 = DIV2 2 = DIV4
5:3	0x0	SC0_V_QUALIFIER: SC0 Vertical Qualifier 0= no vertical qualifier 2= vertical display active 3= 1-line extended vertical display active 4= V Pulse 0 (VP0) 5= 1-line extended V Pulse 0 others= reserved If SC0 is divided by 2 or 4 then it is synchronously reset at the beginning of the horizontal qualifier such that rising edge of SC0 is generated for the first horizontally qualified 'pixel'. In the case where there is no horizontal qualifier start of horizontal display active will be used to generate the synchronous reset. If Initialization Sequence (IS) is enabled on parallel interface then only divide by 1 is allowed for SC0 Clock Divider and SC0 must have vertical and horizontal qualifiers enabled. 0 = NO_VQUAL 1 = RESERVED 2 = VACTIVE 3 = EXT_VACTIVE 4 = VPULSE0 5 = EXT_VPULSE0
2:0	0x0	SC0_H_QUALIFIER: SC0 Horizontal Qualifier 0= disable (regardless of vertical qualifier) 1= no horizontal qualifier (V qualifier only) 2= horizontal display active 3= 1-clock early & extended H display active 4= H Pulse 0 (HP0) 5= 1-clock early & extended H Pulse 0 others= reserved 0 = DISABLE 1 = NO_HQUAL 2 = HACTIVE 3 = EXT_HACTIVE 4 = HPULSE0 5 = EXT_HPULSE0

29.9.51 DC_DISP_DATA_ENABLE_OPTIONS_0

Data Enable options

DE signal is display Data Enable signal which can be used to indicate valid data area and it can be output on LCD_WR_N pin if needed.

This signal can also be used to generate PCS (Parallel mode panel Chip Selecter)

- 0 = De-asserted during both VBlank and HBlank, except for the IS sequence and the gap between IS and the first active line if IS is enabled.
- 1 = De-asserted during VBlank, except for the IS line (whole IS line) if IS is enabled.
- 2 = Always asserted during refresh

Option 0 can be achieved by: DE_SELECT=ACTIVE_IS; DE_CONTROL=NORMAL

Option 1 can be achieved by: DE_SELECT=ACTIVE_IS; DE_CONTROL=ACTIVE_BLANK

Option 2 can be achieved by: DE_SELECT=ACTIVE_BLANK; DE_CONTROL=ACTIVE_BLANK

Offset: 432h | Read/Write: R/W | Reset: 0b00000

Bit	Reset	Description
4:2	0x0	DE_CONTROL: DE (Data Enable) horizontal coverage control 0= 1-pixel clock pulse preceding active line (1-clock DE) 1= LDE active for horizontal display active time (normal DE) 2= LDE starts 1-pixel clock preceding active line but stays high on horizontal display active (early and extended DE) 3= 1-pixel clock early horizontal display active (early DE) 4= DE is active for the whole line, covering both active data and h blank (active and blank) 0 = ONECLK 1 = NORMAL 2 = EARLY_EXT 3 = EARLY 4 = ACTIVE_BLANK
1:0	0x0	DE_SELECT: DE (Data Enable) vertical coverage control 0= DE is generated on every lines (active & blank) 1= DE is generated only for active lines 2= DE is generated for active lines and Initialization sequence (if IS is enabled). DE is also asserted in the time gap between the IS and the first active line. This bit also controls STH for serial display interface in the same manner. 0 = ACTIVE_BLANK 1 = ACTIVE 2 = ACTIVE_IS

29.9.52 DC_DISP_SERIAL_INTERFACE_OPTIONS_0

Serial Display Interface Options

Controls signals for the low-voltage differential serial display interface consists of: SDT, STP and STH signals. SDT and STP are asserted high if current pixel is same as previous pixel; in this case,

SDT is toggled low sometime later but STP is either toggled low at same time as SDT (if next pixel is different than current pixel) or remains high if next pixel is same as current pixel.

When doing pixel comparison, output of dither is used, so pixel comparison depends on the base color (which maybe different than the number of output data bits).

Both SDT and STP are always low (disabled) if the pixel clock divider is 4 or less.

STH is used to indicate the beginning of line and it is asserted high once at the beginning of each line. The STH pulse exact timing width is dependent on the exact mode. STH is generated from Data Enable therefore Data Enable Select bit also

controls STH generation and can be used to generate STH either only for active lines or both for active and blank lines. If STH is sent during blank lines then the blank lines are also transmitted.

Offset: 433h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	STP_CONTROL: STP signal control 0= STP is not OR-ed with H Pulse 2 and vertical blank 1= STP is OR-ed with H Pulse 2 and vertical blank This may be set to 1 when STP needs to be forced high during blank time in which case H Pulse 2 should be programmed when STP needs to be forced high. Vertical blank is the area outside vertical display active. 0 = NORMAL 1 = EXTENDED
6	0x0	STH_DURATION: STH signal duration 0= STH is high for 1 pixel clock in all cases except for 3-bit 2-channel and 6-bit 3-channel where STH is 1.5 pixel clock and for 3-bit 3-channel STH is 3 pixel clocks. 1= STH is high for 2 pixel clock in all cases except for 3-bit 3-channel STH is 4 pixel clocks. 0 = ONE_CLOCK 1 = TWO_CLOCK
5:2	0x0	SDT_STP_DURATION: SDT and STP signal duration 0= 1 shift clock 1= 1 pixel clock 2= 1 pixel clock - 1 shift clock 3= 1 pixel clock - 2 shift clock 4= 1 pixel clock - 3 shift clock 5= 1 pixel clock - 4 shift clock : : : F= 1 pixel clock - 14 shift clock STP active duration is same as SDT if next pixel is not the same as current pixel; else, STP active duration is always 1 pixel clock. Maximum valid setting is pixel clock divider - 1 for pixel clock divider > 4. If pixel clock divider is 4 or less, SDT and STP is always low.
1:0	0x0	SDT_STP_MODE: SDT and STP modes 0= SDT and STP disabled 1= reserved 2= SDT & STP enabled, duplicate data sent 3= SDT & STP enabled, duplicate data not sent 0 = DISABLE 1 = RESERVED 2 = ENABLE_DUP 3 = ENABLE

29.9.53 DC_DISP_LCD_SPI_OPTIONS_0

LCD SPI Interface Options

LCD SPI interface signals consists of:

1. SPI Clock (SCK) which can be output on LCD_SCK pin.

2. SPI Data (SDA) which can be output on LCD_SDOUT pin.

3. Optional SPI Data/Command (SDC) which can be output on LCD_DC0 pin.

4. Main-Display SPI Chip Select (Main SCS_) signal which can be output on LCS_ pin.

5. Sub-Display SPI Chip Select (Sub SCS_) signal which can be optionally output on several pins (see pin output select) - this is optional and it is used only if there is a sub display.

For LCD SPI, pixel data can only be sent to either Main-Display or Sub-Display but not to both.

Main SCS_ or Sub SCS_ signal is always active low and is typically controlled by SPI logic but can also be forced active one line prior to display active (for SIS SPI) and during vertical display active area (for LCD SPI).

Offset: 434h | Read/Write: R/W | Reset: 0b00000

Bit	Reset	Description
4	0x0	LCD_SPI_DIRECTION: LCD SPI Data Direction. Note that data direction does not affect the start byte direction (which is always msb to lsb) and position (always first 8-bit of serial data) for SPI16SB mode. 0 = MSB2LSB 1 = LSB2MSB
3:2	0x0	SPI_CS_CONTROL: LCD SPI Chip Select (SCS_) Control for both IS SPI or LCD SPI 0= Main SCS_ or Sub SCS_ is controlled by LCD SPI or by IS SPI 1= Main SCS_ or Sub SCS_ is controlled by LCD SPI, and depending on LCD SPI Chip Select bit, one of them is forced active for 1-line prior to display active when IS SPI is enabled 2= Main SCS_ or Sub SCS_ is controlled by IS SPI, and depending on LCD SPI Chip Select bit, one of them is forced active during vertical display active area when LCD SPI is enabled 3= Main SCS_ or Sub SCS_, depending on LCD SPI Chip Select bit, is forced active 1-line prior to display active when IS SPI is enabled and also during vertical display active area when LCD SPI is enabled 0 = LCD_IS_SPI 1 = LCD_SPI 2 = IS_SPI 3 = FORCED
1	0x0	LCD_SPI_DC: LCD SPI Data/Command (SDC) 0= SPI Data/Command is low for LCD SPI writes to the display. For PCD6 data format, command byte is sent. 1= SPI Data/Command is high for LCD SPI writes to the display. For PCD6 data format, data byte is sent. 0 = LOW 1 = HIGH
0	0x0	LCD_SPI_CS: LCD SPI Chip Select (SCS_) 0= Send LCD SPI data to Main Display (Main SCS_ is activated) 1= Send LCD SPI data to Sub Display (Sub SCS_ is activated) This bit is also used when SPI Chip Select Control are NOT LCD_IS_SPI to determine either Main SCS_ or Sub SCS_ to be forced active. 0 = MAIN 1 = SUB

29.9.54 DC_DISP_BORDER_COLOR_0

Border Color

Border Color defines the color of areas within the active display area which are outside the defined active windows. This is 24-bit color which is applied after blending.

Offset: 435h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	BORDER_COLOR_B: Blue Border Color
15:8	X	BORDER_COLOR_G: Green Border Color
7:0	X	BORDER_COLOR_R: Red Border Color

29.9.55 DC_DISP_COLOR_KEY0_LOWER_0

Color Key 0 Lower value

Color Key 0 and Color Key 1

Two ranges of color key are defined and they are common for all windows because it is expected that typically only one window will have color key enabled. Because there are two sets of color 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: 436h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	COLOR_KEY0_L_B: Color Key 0 Blue (U) Lower value
15:8	X	COLOR_KEY0_L_G: Color Key 0 Green (Y) Lower value
7:0	X	COLOR_KEY0_L_R: Color Key 0 Red (V) Lower value

29.9.56 DC_DISP_COLOR_KEY0_UPPER_0

Color Key 0 Upper value

Offset: 437h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	COLOR_KEY0_U_B: Color Key 0 Blue (U) Upper value
15:8	X	COLOR_KEY0_U_G: Color Key 0 Green (Y) Upper value
7:0	X	COLOR_KEY0_U_R: Color Key 0 Red (V) Upper value

29.9.57 DC_DISP_COLOR_KEY1_LOWER_0

Color Key 1 Lower value

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

Bit	Reset	Description
23:16	X	COLOR_KEY1_L_B: Color Key 1 Blue (U) Lower value
15:8	X	COLOR_KEY1_L_G: Color Key 1 Green (Y) Lower value
7:0	X	COLOR_KEY1_L_R: Color Key 1 Red (V) Lower value

29.9.58 DC_DISP_COLOR_KEY1_UPPER_0

Color Key 1 Upper value

Offset: 439h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	COLOR_KEY1_U_B: Color Key 1 Blue (U) Upper value
15:8	X	COLOR_KEY1_U_G: Color Key 1 Green (Y) Upper value
7:0	X	COLOR_KEY1_U_R: Color Key 1 Red (V) Upper value

29.9.59 DC_DISP_CURSOR_FOREGROUND_0

Cursor Foreground color

Class: Hardware Cursor

Hardware cursor is supported for 32x32 or for 64x64 2-bpp cursor.

Cursor start address is aligned to 1 KB boundary. All cursor registers except for cursor foreground and background colors are triple buffered.

GENERAL_UPDATE controls ASSEMBLY->ARM latching, GENERAL_ACT_REQ controls ARM->ACTIVE latching.

Cursor scaling and flipping are not implemented so this must be done by software if needed. Cursor H/V positions are signed number with respect to one of the display windows or with respect to upper left position of display active area as specified by cursor clipping parameter which also determines cursor clipping boundary. If cursor position is with respect to one of the display window and the corresponding display window is disabled then cursor will also be disabled.

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

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

29.9.60 DC_DISP_CURSOR_BACKGROUND_0

Cursor Background color

Offset: 43dh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

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

29.9.61 DC_DISP_CURSOR_START_ADDR_0

Cursor Start Address

Offset: 43eh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
29:28	X	CURSOR_CLIPPING: Cursor Clipping Select 0 = DISPLAY : 00= display 1 = WA : 01= window A 2 = WB : 10= window B 3 = WC : 11= window C
24	X	CURSOR_SIZE: Cursor Size 0= 32x32 1= 64x64 0 = C32X32 1 = C64X64
21:0	X	CURSOR_START_ADDR: Cursor Start Address bits 25:10

29.9.62 DC_DISP_CURSOR_START_ADDR_NS_0

Shadow of Cursor Start Address

Offset: 43fh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
29:28	X	CURSOR_CLIPPING_NS: Cursor Clipping Select 0 = DISPLAY : 00= display 1 = WA : 01= window A 2 = WB : 10= window B 3 = WC : 11= window C
24	X	CURSOR_SIZE_NS: Cursor Size 0= 32x32 1= 64x64 0 = C32X32 1 = C64X64
21:0	X	CURSOR_START_ADDR_NS: Cursor Start Address bits 25:10

29.9.63 DC_DISP_CURSOR_POSITION_0

Cursor Position

Cursor position is with respect to top-left corner of display active area, or window A, or window B, or window C as specified cursor clipping parameter.

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

Bit	Reset	Description
29:16	X	V_CURSOR_POSITION: V cursor position (signed)
13:0	X	H_CURSOR_POSITION: H cursor position (signed)

29.9.64 DC_DISP_CURSOR_POSITION_NS_0

Shadow of Cursor Position

Offset: 441h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
29:16	X	V_CURSOR_POSITION_NS: V cursor position (signed)
13:0	X	H_CURSOR_POSITION_NS: H cursor position (signed)

29.9.65 DC_DISP_INIT_SEQ_CONTROL_0

Initialization Sequence Control

Class: Initialization Sequence (IS)

Display initialization sequence may have to be written to the display if the display has built-in frame buffer. This initialization sequence is used typically to reinitialize the display buffer start address and maybe needed once per frame (frame initialization sequence) and/or once per line (line initialization sequence).

Frame initialization sequence is sent during the horizontal active time of the line just before the first active display line. Line initialization sequence is currently NOT supported.

Initialization sequence can be done through parallel LCD interface or through SPI serial interface. Software is responsible in making sure that the active line time is sufficient to send initialization sequence.

For parallel interface initialization, the signals used as chip selects (typically these are one of the vertical signals) must be programmed to be active one line just before the first active display line. Also SC0/SC1 clock divider must be programmed to divide by 1 if initialization sequence is enabled.

Offset: 442h | Read/Write: R/W | Reset: 0bxxxxxxxxxx00

Bit	Reset	Description
11:8	X	FRAME_INIT_SEQ_CYCLES: Frame Initialization Sequence Cycles This parameter specifies the number of frame initialization sequence cycles to send. If programmed to 0, there is no frame initialization cycle generated.
7	X	INIT_SEQ_DC_CONTROL: Initialization Sequence DC Pin This bit is used only for parallel initialization sequence and it controls how data/command is added to the vertical signal selected by -=PS_=-INIT_SEQ_DC_SIGNAL 0= parallel IS DC is inverted and then AND-ed to the vertical signal 1= parallel IS DC is OR-ed to the vertical signal
6:4	X	INIT_SEQ_DC_SIGNAL: Frame Initialization Sequence DC Pin This parameter is used only for parallel initialization sequence and it specifies which signal carries the data/command signal. 0= parallel IS DC signal is not needed 1= parallel IS DC on Vertical Sync 2= parallel IS DC on Vertical Pulse 0 3= parallel IS DC on Vertical Pulse 1 4= parallel IS DC on Vertical Pulse 2 5= parallel IS DC on Vertical Pulse 3 other= reserved 0 = NODC 1 = VSYNC 2 = VPULSE0 3 = VPULSE1 4 = VPULSE2 5 = VPULSE3
1	0x0	INIT_SEQUENCE_MODE: Initialization Sequence Mode 0= Send init sequence through parallel LCD interface 1= Send init sequence through SPI serial interface 0 = PLCD_INIT 1 = SPI_INIT

Bit	Reset	Description
0	0x0	SEND_INIT_SEQUENCE: Send Initialization Sequence (IS) 0 = DISABLE 1 = ENABLE

29.9.66 DC_DISP_SPI_INIT_SEQ_DATA_A_0

SPI Init Sequence Write Data A

For parallel initialization sequence there are two possible data widths: 9 bits or 18 bits.

If parallel IS is selected, the number of bits per cycle depend on the DISP_DATA_FORMAT register programming. 18-bit parallel IS cycles are performed for 1-pixel/1-clock parallel interface (DF1P1C). 9-bit parallel IS cycles are performed for non 1-pixel/1-clock parallel interface.

Parallel IS cycles must be completed prior to the end of horizontal active of the line where IS cycles are sent. If all the cycles have been completed prior to the end of horizontal active, control signals are held inactive and last output data is held till end of horizontal active. For 9-bit parallel initialization sequence, the data is output in either LCD_D[8:0] pins or LCD_D[17:9] pins depending on display data alignment.

For serial initialization sequence using SPI (IS SPI) there are six possible data widths: 8 bits, 9 bits, 12 bits, 16 bits, 16 bits data plus start byte (24 bits), 18 bits, or 24 bits.

Parameters in SPI_CONTROL register and SPI_START_BYTE register is also used for serial initialization sequence using SPI.

Serial IS cycles must also be completed prior to the end of horizontal active of the line where initialization cycles are sent. The programmer needs to make sure that register programming is such that this is true. If all the cycles have been completed prior to the end of horizontal active, SPI signals will be forced inactive until the next SPI cycles.

The following shows how initialization sequence data bits are used when sending initialization sequence:

For 9-bit parallel initialization - up to 10 initialization cycles can be done:

Init cycle	1	2	3	4	5	6	7	8	9	10
Data	8-0	17-9	26-18	35-27	44-36	53-45	62-54	71-63	80-72	89-81
LCD_PCLK enable	90	93	96	99	102	105	108	111	114	117
LCD_WR_N enable	91	94	97	100	103	106	109	112	115	118
data/command	92	95	98	101	104	107	110	113	116	119

For 18-bit parallel initialization - up to 6 initialization cycles can be done:

Init cycle	1	2	3	4	5	6
Data	17-0	35-18	53-36	71-54	89-72	107-90
LCD_PCLK enable	108	111	114	117	120	123
LCD_WR_N enable	109	112	115	118	121	124
data/command	110	113	116	119	122	125

For serial initialization using SPI, main display SPI chip select (Main SCS_) is always output on LCS_ pin while sub display SPI chip select (Sub SCS_) can be optionally output on several pins (see pin output select definition).

Initialization cycle through SPI interface can only be sent to either main or sub display but not to both and the selection bits are specified in the tables below.

Note that 0 indicates main display initialization and 1 indicates sub display initialization.

For 8-bit SPI initialization - up to 12 initialization cycles can be done:

Init cycle	1	2	3	4	5	6	7	8	9	10	11	12
Data	7-0	15-8	23-16	31-24	39-32	47-40	55-48	63-56	71-64	79-72	87-80	95-88
Main/Sub SCS_	96	98	100	102	104	106	108	110	112	114	116	118
SDC	97	99	101	103	105	107	109	111	113	115	117	119

For 12-bit SPI initialization - up to 9 initialization cycles can be done:

Init cycle	1	2	3	4	5	6	7	8	9
Data	11-0	23-12	35-24	47-36	59-48	71-60	83-72	95-84	107-96
Main/Sub SCS_	109	111	113	115	117	119	121	123	125
LCD_DC0	110	112	114	116	118	120	122	124	126

For 16-bit SPI initialization - up to 7 initialization cycles can be done:

Init cycle	1	2	3	4	5	6	7
Data	15-0	31-16	47-32	63-48	79-64	95-80	111-96
Main/Sub SCS_	112	114	116	118	120	122	124
SDC	113	115	117	119	121	123	125

For 18-bit SPI initialization - up to 6 initialization cycles can be done:

Init cycle	1	2	3	4	5	6
Data	17-0	35-18	53-36	71-54	89-72	107-90
Main/Sub SCS_	108	110	112	114	116	118
SDC	109	111	113	115	117	119

For 24-bit SPI initialization - up to 4 initialization cycles can be done:

Init cycle	1	2	3	4
Data	23-0	47-24	71-48	95-72
Main/Sub SCS_	96	98	100	102
SDC	97	99	101	103

Offset: 443h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SPI_INIT_SEQ_DATA_A: SPI Init Sequence Write Data bits 31-0

29.9.67 DC_DISP_SPI_INIT_SEQ_DATA_B_0

SPI Init Sequence Write Data B

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

Bit	Reset	Description
31:0	X	SPI_INIT_SEQ_DATA_B: SPI Init Sequence Write Data bits 63-32

29.9.68 DC_DISP_SPI_INIT_SEQ_DATA_C_0

SPI Init Sequence Write Data C

Offset: 445h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SPI_INIT_SEQ_DATA_C: SPI Init Sequence Write Data bits 95-64

29.9.69 DC_DISP_SPI_INIT_SEQ_DATA_D_0

SPI Init Sequence Write Data D

Offset: 446h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SPI_INIT_SEQ_DATA_D: SPI Init Sequence Write Data bits 127-96

29.9.70 DC_DISP_DC_MCCIF_FIFOCTRL_0

Memory Client Interface FIFO Control Register

The registers below enable optimizing of the synchronization timing in the memory client asynchronous FIFOs. When they can be used depends on the client and memory controller clock ratio. The RDMC_RDFAST/RDCL_RDFAST fields can increase power consumption if the asynchronous FIFO is implemented as a real ram.

There is no power impact on latch-based FIFOs. Flip-flop-based FIFOs do not use these fields.

Recommended Settings

- Client writing to FIFO, memory controller reading from FIFO
  - MCCLK_FREQ <= CLIENTCLK_FREQ

    You can enable both RDMC_RDFAST and WRCL_CLLE2X. If one of the FIFOs is a real ram and power is a concern, you should avoid enabling RDMC_RDFAST.
  - CLIENTCLK_FREQ < MCCLK_FREQ <= 2 * CLIENTCLK_FREQ

    You can enable RDMC_RDFAST or WRCL_MCLE2X, but because the client clock is slower, you should enable only WRCL_MCLE2X.
  - 2 * CLIENTCLK_FREQ < MCCLK_FREQ

    You can only enable RDMC_RDFAST. If one of the FIFOs is a real ram and power is a concern, you should avoid enabling RDMC_RDFAST.
- Memory controller writing to FIFO, client reading from FIFO
  - CLIENTCLK_FREQ <= MCCLK_FREQ

    You can enable both RDCL_RDFAST and WRMC_CLLE2X. If one of the FIFOs is a real ram and power is a concern, you should avoid enabling RDCL_RDFAST.
  - MCCLK_FREQ < CLIENTCLK_FREQ <= 2 * MCCLK_FREQ

    You can enable RDCL_RDFAST or WRMC_CLLE2X, but because the memory controller clock is slower, you should enable only WRMC_CLLE2X.
  - - 2 * MCCLK_FREQ < CLIENTCLK_FREQ

You can only enable RDCL_RDFAST. If one of the FIFOs is a real ram and power is a concern, you should avoid enabling RDCL_RDFAST.

Offset: 480h | Read/Write: R/W | Reset: 0b0000

Bit	Reset	Description
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

29.9.71 DC_DISP_MCCIF_DISPLAY0A_HYST_0

Memory Client Hysteresis Control Register

This register exists only for clients with hysteresis. HYST_EN can be used to turn on or off the hysteresis logic. HYST_REQ_TH is the threshold of pending requests required before allowing them to pass through (overridden after HYST_REQ_TM cycles). Hysteresis logic will stop holding request after (1<< HYST_TM) cycles (this should not normally have to be used).

Deep hysteresis is a second level of hysteresis on a longer time-frame. DHYST_TH is the size of the read burst (requests are held until there is space for the entire burst in the return data FIFO). During a burst period, if there are no new requests after DHYST_TM cycles, then the burst is terminated early.

Offset: 481h | Read/Write: R/W | Reset: 0b11001111010000000001111100011111

Bit	Reset	Description
31	ENABLE	CBR_DISPLAY0A2MC_HYST_EN: 1 = ENABLE 0 = DISABLE
30:28	0x4	CBR_DISPLAY0A2MC_HYST_REQ_TH
27:24	0xf	CBR_DISPLAY0A2MC_HYST_TM
23:16	0x40	CBR_DISPLAY0A2MC_DHYST_TH
15:8	0x1f	CBR_DISPLAY0A2MC_DHYST_TM
7:0	0x1f	CBR_DISPLAY0A2MC_HYST_REQ_TM

29.9.72 DC_DISP_MCCIF_DISPLAY0B_HYST_0

Memory Client Hysteresis Control Register

This register exists only for clients with hysteresis. HYST_EN can be used to turn on or off the hysteresis logic. HYST_REQ_TH is the threshold of pending requests required before allowing them to pass through (overridden after

HYST_REQ_TM cycles). Hysteresis logic will stop holding request after (1<<HYST_TM) cycles (this should not normally have to be used).

Deep hysteresis is a second level of hysteresis on a longer time-frame. DHYST_TH is the size of the read burst (requests are held until there is space for the entire burst in the return data FIFO). During a burst period, if there are no new requests after DHYST_TM cycles, then the burst is terminated early.

Offset: 482h | Read/Write: R/W | Reset: 0b11001111000010000001111100011111

Bit	Reset	Description
31	ENABLE	CBR_DISPLAY0B2MC_HYST_EN: 1 = ENABLE 0 = DISABLE
30:28	0x4	CBR_DISPLAY0B2MC_HYST_REQ_TH
27:24	0xf	CBR_DISPLAY0B2MC_HYST_TM
23:16	0x8	CBR_DISPLAY0B2MC_DHYST_TH
15:8	0x1f	CBR_DISPLAY0B2MC_DHYST_TM
7:0	0x1f	CBR_DISPLAY0B2MC_HYST_REQ_TM

29.9.73 DC_DISP_MCCIF_DISPLAY0C_HYST_0

Memory Client Hysteresis Control Register

This register exists only for clients with hysteresis. HYST_EN can be used to turn on or off the hysteresis logic. HYST_REQ_TH is the threshold of pending requests required before allowing them to pass through (overridden after HYST_REQ_TM cycles). Hysteresis logic will stop holding request after (1<<HYST_TM) cycles (this should not normally have to be used and is only a WAR for unexpected hangs).

Deep hysteresis is a second level of hysteresis on a longer time-frame. DHYST_TH is the size of the read burst (requests are held until there is space for the entire burst in the return data FIFO). During a burst period, if there are no new requests after DHYST_TM cycles, then the burst is terminated early.

Offset: 483h | Read/Write: R/W | Reset: 0b11001111000010000001111100011111

Bit	Reset	Description
31	ENABLE	CBR_DISPLAY0C2MC_HYST_EN: 1 = ENABLE 0 = DISABLE
30:28	0x4	CBR_DISPLAY0C2MC_HYST_REQ_TH
27:24	0xf	CBR_DISPLAY0C2MC_HYST_TM
23:16	0x8	CBR_DISPLAY0C2MC_DHYST_TH
15:8	0x1f	CBR_DISPLAY0C2MC_DHYST_TM
7:0	0x1f	CBR_DISPLAY0C2MC_HYST_REQ_TM

29.9.74 DC_DISP_MCCIF_DISPLAY1B_HYST_0

Memory Client Hysteresis Control Register

This register exists only for clients with hysteresis. HYST_EN can be used to turn on or off the hysteresis logic. HYST_REQ_TH is the threshold of pending requests required before allowing them to pass through (overridden after HYST_REQ_TM cycles). Hysteresis logic will stop holding request after (1<<HYST_TM) cycles (this should not normally have to be used).

Deep hysteresis is a second level of hysteresis on a longer time-frame. DHYST_TH is the size of the read burst (requests are held until there is space for the entire burst in the return data FIFO). During a burst period, if there are no new requests after DHYST_TM cycles, then the burst is terminated early.

Offset: 484h | Read/Write: R/W | Reset: 0b11001111000010000001111100011111

Bit	Reset	Description
31	ENABLE	CBR_DISPLAY1B2MC_HYST_EN: 1 = ENABLE 0 = DISABLE
30:28	0x4	CBR_DISPLAY1B2MC_HYST_REQ_TH
27:24	0xf	CBR_DISPLAY1B2MC_HYST_TM
23:16	0x8	CBR_DISPLAY1B2MC_DHYST_TH
15:8	0x1f	CBR_DISPLAY1B2MC_DHYST_TM
7:0	0x1f	CBR_DISPLAY1B2MC_HYST_REQ_TM

29.9.75 DC_DISP_DAC_CRT_CTRL_0

CRT Control Register

Control Registers for CRT Mode (including CYA bits). Control registers for triple DAC/CRT operation (display2tvdac signals)

A register outside of display, TVDACCCONFIG, controls which source among display/displayb/tvo goes to TVDAC.

Offset: 4c0h | Read/Write: R/W | Reset: 0b0x0x0

Bit	Reset	Description
4	0x0	NOTBLANK_SELECT: Selects the source for display2tvdac_notblank 0: notblank = d_active[10] (i.e. data_enable) 1: notblank = (lvp[1] & lhp[1]) 0 = DE 1 = LVP1_LHP1
2	0x0	SYNC_SELECT: Selects the source for display2tvdac_[hv]sync 0 = VSYNC_HSYNC 1 = LVP0_LHP0
0	0x0	OVERRIDE_NOTBLANK: If enabled, output display2tvdac_NOTBLANK is tied to 1 0= disable 1= enable 0 = DISABLE 1 = ENABLE

29.9.76 DC_DISP_DISP_MISC_CONTROL_0

Miscellaneous controls, including CYA features.

Offset: 4c1h | Read/Write: R/W | Reset: 0b10

Bit	Reset	Description
1	0x1	UF_LINE_FLUSH: Enable underflow line flush, a.o.t 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

29.10 Window A (WINC_A) Registers

These registers control window A parameters.

> **Note:** There are three copies of these registers for window A, B, and C. Window A, B, and C support different features

Window A: color palette, digital vibrance

Window B: color palette, digital vibrance, color space conversion, horizontal/vertical filtering

Window C: color palette, digital vibrance, color space conversion, horizontal filtering. The registers under DC_WINC are usually not shadowed.

29.10.1 DC_WINC_A_COLOR_PALETTE_0

This is used for palletized data format (color depth of 8-bpp or less) or for gamma correction for non-palletized data formats (color depth of more than 8-bpp).

Each window has its own color palette which consists of three 256x8 register file which can be written by host and indexed (read) by the window.

For palletized data format less than 8-bpp the pixel data is aligned to least significant bits of the palette index (address) and the remaining upper bits are filled with the corresponding bits of the Palette Color Extension. For example, for 4-bpp mode, the pixel data occupies bits 3-0 of the palette index and bits 7-4 of the palette index are set to bits 7-4 of the Palette Color Extension.

Note that host read is assumed to be not needed - software can cache the color palette in system memory.

This is an array of 256 identical register entries; the register fields below apply to each entry.

Window A Color Palette

Offset: 500h..5ffh | Read/Write: WO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	A_COLOR_PALETTE_B: Blue Color Palette
15:8	X	A_COLOR_PALETTE_G: Green Color Palette
7:0	X	A_COLOR_PALETTE_R: Red Color Palette

29.10.2 DC_WINC_A_PALETTE_COLOR_EXT_0

Palette extension for 1-bpp, 2-bpp, and 4-bpp. These bits provide the upper most significant bits for indexing the color palette. Supported for window A only. XXX should ifdef for window A, but currently window B spec is assumed to be a superset.

Window A Palette Color Extension

Offset: 600h | Read/Write: R/W | Reset: 0bxxxxxxx

Bit	Reset	Description
7:1	X	A_PALETTE_COLOR_EXT: Window A Palette Color Extension bits 7-1 are used for 1-bpp mode bits 7-2 are used for 2-bpp mode bits 7-4 are used for 4-bpp mode

Horizontal Scaling Filter Coefficients

Horizontal scaling filter is a 6-tap filter with 4-bit positional phase.

- Coefficients 0 and 5 are 3-bit signed value ranging from -4 to 3.
- Coefficients 1 and 4 are 5-bit signed value ranging from -16 to 15.
- Coefficients 2 and 3 are 8-bit unsigned value ranging from 0 to 128.
- Coefficient 0 is the multiplier for the earliest pixel (P0) in the group of 6-pixel and
- Coefficient 5 is the multiplier for the latest pixel (P5) in the group. The output pixel positional phase is defined as centered in P2 if the positional phase is 0 or proportionally in between P2 and P3 if the positional phase is larger than 0.

Sum of all coefficients for each phase should be 128 typically and software should never program the the sum of all coefficients for a phase to be more than 128.

For each horizontal positional phase, the 6 filter coefficients require 32 reg bits.

Note that color value ranges from 0 to 255 for Y, R, G, B and -128 to 127 for U and V.

Color Space Conversion coefficients

The CSC can be used for YUV to RGB conversion with brightness and hue/saturation control. The CSC can only be enabled for window A controlled by CSC_ENABLE register bits.

For Y color, the Y offset is applied first and saturation (clipping) is performed immediately after the Y offset is applied.

R = sat(KYRGB * sat(Y + YOF) + KUR * U + KVR * V)

G = sat(KYRGB * sat(Y + YOF) + KUG * U + KVG * V)

B = sat(KYRGB * sat(Y + YOF) + KUB * U + KVB * V)

Saturation and rounding is performed in the range of 0 to 255 for the above equations.

Typical values are:

YOF = -16.000, KYRGB = 1.1644

KUR = 0.0000, KVR = 1.5960

KUG = -0.3918, KVG = -0.8130

KUB = 2.0172, KVB = 0.0000

KUR and KVB are typically 0.0000 but they may be programmed non-zero for hue rotation.

The CSC can also take RGB input, in which case YOF, KVB, KUG, KUR should be programmed to 0 and KYRGB will be forced to 0 by the hardware for generating R and B. KYRGB will not be forced to 0 for generating G. KVR, KYRGB, and KUB can be programmed to 1.0 or used as gain control for R, G, B correspondingly.

Note that color value ranges from 0 to 255 for Y, R, G, B and -128 to 127 for U and V.

Vertical scaling filter coefficients

Vertical scaling filter is a 2-tap filter with 4-bit positional phase.

Coefficients 0 and 1 are 8-bit unsigned value ranging from 0 to 128.

Coefficient 0 is the multiplier for the earlier pixel (P0) in the group of 2-pixel and coefficient 1 is the multiplier for the later pixel (P1) in the group. The output pixel positional phase is defined as centered in P0 if the positional phase is 0 or proportionally in between P0 and P1 if the positional phase is larger than 0.

Sum of all coefficients for each phase should be 128 typically therefore coefficient 1 can be calculated from (1 - coefficient 0) and therefore only coefficient 0 is programmed.

For each vertical positional phase, the filter coefficient requires 8 reg bits. Note that color value ranges from 0 to 255 for Y, R, G, B and -128 to 127 for U and V.

The registers under DC_WIN are double buffered.

29.10.3 DC_WIN_A_WIN_OPTIONS_0

Window A Options

Class: Display Window Settings

Display Window A parameters

Offset: 700h | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxx0xxxxxxxxxxxxxxx

Bit	Reset	Description
30	0x0	A_WIN_ENABLE: Window A Window enable 0 = DISABLE 1 = ENABLE
20	X	A_DV_ENABLE: Window A Digital Vibrance Enable 0 = DISABLE 1 = ENABLE
16	0x0	A_CP_ENABLE: Window A Color Palette Enable This controls the color palette and should be enabled for palletized color modes. For non-palletized color modes, the color palette can be enabled for gamma correction. 0 = DISABLE 1 = ENABLE
6	X	A_COLOR_EXPAND: Window A 12/15/16/18-to-24 bpp color expansion This bit should be enabled only for 12-bpp B4G4R4A4, 15-bpp B5G5R5A, 16-bpp B5G6R5, 18-bpp B6G6R6 color modes. If enabled the color conversion is performed prior to horizontal scaling. 0 = DISABLE 1 = ENABLE
2	X	A_V_DIRECTION: Window A Vertical (Y) drawing Direction 0 = INCREMENT 1 = DECREMENT

Bit	Reset	Description
0	X	A_H_DIRECTION: Window A Horizontal (X) drawing Direction 0 = INCREMENT 1 = DECREMENT

29.10.4 DC_WIN_A_BYTE_SWAP_0

Window A Byte Swap

Offset: 701h | Read/Write: R/W | Reset: 0bxx

Bit	Reset	Description
1:0	X	A_BYTE_SWAP: Window A Byte Swap This controls byte swap of frame data read from memory prior to any data processing in the display module. 00= no byte swap (3 2 1 0) 01= byte swap for each 2-byte word (2 3 0 1) 10= byte swap for each 4-byte word (0 1 2 3) 11= word swap for each 4-byte word (1 0 3 2) 0 = NOSWAP 1 = SWAP2 2 = SWAP4 3 = SWAP4HW

29.10.5 DC_WIN_A_BUFFER_CONTROL_0

Window A Buffer Control

Offset: 702h | Read/Write: R/W | Reset: 0b000

Bit	Reset	Description
2:0	0x0	A_BUFFER_CONTROL: Window A Buffer Control 0= Host (software) controlled 1= Video Input controlled 2= Encoder Pre-Processor controlled 3= MPEG Encoder controlled 4= StretchBLT or 2D other= reserved If window buffer selection is not controlled by host (software) then buffer start indexes are sent by the respective module specified by this parameter, and in this case, the buffer start address registers are used to specify frame stride and buffer offset for the calculated start address. 0 = HOST 1 = VI 2 = EPP 4 = SB2D 3 = MPEGE

29.10.6 DC_WIN_A_COLOR_DEPTH_0

Window A Color Depth For YCbCr data format, Cb and Cr are 8-bit unsigned values. For YUV data format, U and V are 8-bit signed values.

YCbCr422R is similar to YCbCr422P but the Cb and Cr are shared vertically.

YUV422R is similar to YUV422P but the U and V are shared vertically.

YCbCr422RA is same as YCbCr422R in memory and YUV422RA is same as YUV422R in memory but while reading from memory, for YCbCr422RA and YUV422RA, chroma averaging is applied for each pixel pair so that they can be processed as YUV422 by the display pipeline.

For YCbCr422R and YUV422R, every other chroma pixels are not used (discarded) by the display pipeline. B6x2G6x2R6x2A8 is similar to B8G8R6A8 but with the 2 lsb zeroed out.

R6x2G6x2B6x2A8 is similar to R8G8B6A8 but with the 2 lsb zeroed out.

Offset: 703h | Read/Write: R/W | Reset: 0bxxxxx

Bit	Reset	Description
4:0	X	A_COLOR_DEPTH: Window A Color Depth Supported color depths are: P1 = 1-bpp (palletized) P2 = 2-bpp (palletized) P4 = 4-bpp (palletized) P8 = 8-bpp (palletized) B4G4R4A4 = 12-bpp B4G4R4 B5G5R5A = 15-bpp B5G5R5 AB5G5R5 = 15-bpp B5G5R5 B5G6R5 = 16-bpp B5G6R5 B8G8R8A8 = 32-bpp B8G8R8A8 R8G8B8A8 = 32-bpp R8G8B8A8 B6x2G6x2R6x2A8 = 32-bpp B6G6R6A8 R6x2G6x2B6x2A8 = 32-bpp R6G6B6A8 0 = P1 1 = P2 2 = P4 3 = P8 4 = B4G4R4A4 5 = B5G5R5A 6 = B5G6R5 7 = AB5G5R5 12 = B8G8R8A8 13 = R8G8B8A8 14 = B6x2G6x2R6x2A8 15 = R6x2G6x2B6x2A8 16 = YCbCr422 17 = YUV422 18 = YCbCr420P 19 = YUV420P 20 = YCbCr422P 21 = YUV422P 22 = YCbCr422R 23 = YUV422R 24 = YCbCr422RA 25 = YUV422RA

29.10.7 DC_WIN_A_POSITION_0

Window A Position

This register defines H position and size of Window A after scaling (if there is any)

Offset: 704h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	A_V_POSITION: Window A V Position This is specified with respect to the top edge of active display area.
12:0	X	A_H_POSITION: Window A H Position This is specified with respect to the left edge of active display area.

29.10.8 DC_WIN_A_SIZE_0

Window A Size

This register defines V position and size of Window A after scaling (if there is any)

Note: programming on window size should guarantee the whole window is inside the active area, otherwise extra rows/columns will be fetched and discarded which affects performance.

Offset: 705h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	A_V_SIZE: Window A V Size (lines) This is the vertical size after scaling.
12:0	X	A_H_SIZE: Window A H Size (pixels) This is the horizontal size after scaling.

29.10.9 DC_WIN_A_PRESCALED_SIZE_0

Window A Pre-scaled Size

This register defines Window A pre-scaled size.

The H pre-scaled size is needed to determine how many bytes to fetch from memory per line and this parameter must be programmed exactly as needed taking into account the scaling factor. For planar YUV or YCbCr data formats, this parameter refer to the H pre-scaled of the Y plane.

The total number of lines to be fetched from memory is determined by post-scale V size but V pre-scaled size is needed to 'clamp' the last valid line if the vertical DDA is exactly or slightly beyond the specified V pre-scaled size.

Design Note: H pre-scaled size ideally should be in terms of pixel but then hardware needs to convert this precisely to bytes to determine the amount of data to request from memory.

This could be a risky calculation - maybe this should be made optional on whether we use internal hardware to calculate or left it to software to calculate.

Offset: 706h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	A_V_PRESCALED_SIZE: Window A V Pre-scaled Size (lines) In 420P/422R/422RA formats, it must be even.
14:0	X	A_H_PRESCALED_SIZE: Window A H Pre-scaled Size (bytes) In 420P and 422P formats, it must be even.

29.10.10 DC_WIN_A_H_INITIAL_DDA_0

Window A H Initial DDA

Design Note: the first pixel of pre-scaled image is always used to output the first pixel so essentially this is the same as forcing the H Initial DDA integer portion to 1 initially even though user typically programs this to 0. If it makes the implementation easier, it is possible to force software to program the Initial DDA integer portion to 1. Similarly with the V Initial DDA.

Offset: 707h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	A_H_INITIAL_DDA: Window A H Initial DDA (4.12) This is typically programmed to 0.0

29.10.11 DC_WIN_A_V_INITIAL_DDA_0

Window A V Initial DDA

Offset: 708h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	A_V_INITIAL_DDA: Window A V Initial DDA (4.12) This is typically programmed to 0.0 for both non-interlaced and interlaced sources.

29.10.12 DC_WIN_A_DDA_INCREMENT_0

Window A DDA Increment

DDA increment is typically calculated by dividing (Pre-scaled size in pixels - 1) by(Post-scaled size in pixels - 1). The result should be round-ed up and expressed as 4.12 format (4-bit integer and 12-bit fraction). For non-filtered image this value can be slightly larger so that it is not missing the last row/column. Reference programming values (H and V should be calculated separately depending on filter on/off and sizes):

- Filter on: min(round((prescaled_size_in_pixels - 1) * 0x1000 / (post_scaled_size_in_pixels - 1)), MAX)

- Filter off: min(round(prescaled_size_in_pixels * 0x1000 / (post_scaled_size_in_pixels - 1) - 0.5), MAX)

Where the value of MAX is as follows:

For V_DDA_INCREMENT: 15.0 (0xF000)

For H_DDA_INCREMETN: 4.0 (0x4000) for 2 Bytes/pix formats.

8.0 (0x8000) for 4 Bytes/pix formats.

They are theoretically the biggest values that guarantees not displaying beyond an image boundary. If the DDA increment is less than 1.0 then image will be up-scaled and if DDA increment is more than 1.0 then image will be down-scaled.

Offset: 709h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	A_V_DDA_INCREMENT: Window A Vertical DDA Increment (4.12) This should be set to 1.0 if there is no scaling. Maximum value is 15.0 regardless of the number of bytes per pixel.
15:0	X	A_H_DDA_INCREMENT: Window A Horizontal DDA Increment (4.12) This should be set to 1.0 if there is no scaling. The maximum value for downscaling depends on the number of bytes per pixel. For 4-byte/pixel modes (32-bpp) the maximum value is 4.0 and for all other modes the maximum value is 8.0.

29.10.13 DC_WIN_A_LINE_STRIDE_0

Window A Line Stride

Offset: 70ah | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	A_LINE_STRIDE: Window A Line Stride This is stride (in bytes) for all non-planar data formats. If the memory surface is tiled, the stride needs to be a multiple of 16. If H_DIRECTION of window A is set to DECREMENT, the stride also needs to be a multiple of 16. For planar YUV or YCbCr data formats, this is stride (in bytes) for the luma plane with the restriction that it must be multiples of 8 (16 if tiled or in horizontal flipping) For tiled surface this value may affect starting address of a window. Refer to the comment of START_ADDR for detail.

29.10.14 DC_WIN_A_BUF_STRIDE_0

Window A Buffer stride

Offset: 70bh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	A_BUF_STRIDE: Window A Buffer stride Buffer stride is used to calculate the buffer addresses when the window is triggered by non-host modules. Refer to the comment of of START_ADDR for programming guide. For YUV planar pixel format, this specifies buffer stride for the Y plane. The value is in bytes.

29.10.15 DC_WIN_A_BUFFER_ADDR_MODE_0

Memory Controller Tiling definitions

Offset: 70dh | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
0	0x0	A_TILE_MODE: Window A Memory surface tiling mode For YUV planar pixel format, this specifies tiling mode for the Y plane. 0 = LINEAR 1 = TILED

29.10.16 DC_WIN_A_DV_CONTROL_0

Window A Digital Vibrance Control

If enabled, Digital Vibrance is applied after H and V scaling and after color palette or color space conversion logic but before color keying multiplexer and before cursor multiplexer.

- After DV, new R = R + (2R - G - B) * FR, where FR is fraction from 0 to 7/8
- After DV, new G = G + (2G - R - B) * FG, where FG is fraction from 0 to 7/8
- After DV, new B = B + (2B - R - G) * FB, where FB is fraction from 0 to 7/8

Offset: 70eh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxx

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

29.10.17 DC_WIN_A_BLEND_NOKEY_0

Blend Control for this window areas where color key is enabled but the pixel color is not within the color key range (color key not match). This is valid for all overlapping condition but only if there is no overlap with other window with higher priority color key enabled and color key not match.

Class: Display Color Keying and Blending

Color keying and blending of the display windows are done prior to cursor blending.

Cursor always goes on top of the blended windows. Blending is controlled independently on each possible overlap area of the display windows. If 3 windows are enabled there are 7 possible overlap area combinations. For every window in each overlap area combination, color key can be disabled or enabled. Also, for every window in each overlap area combination, there is a corresponding window blend control parameter and a window blend weight parameter. The window blend control parameter is always effective but the window blend weight is not always used. The window blend weight can also be derived from pixel alpha value or from the reverse of other overlapping windows weight.

Color keying has the highest priority for display window blending. Color key consists of a range of color which is searched independently for each windows. If more than 1 windows color keys are enabled then Window A color key has the highest priority, followed by

Window B color key, and then followed by Window C color key. Two sets of color key range (Color Key 0 and Color Key 1) can be defined and they are shared for all windows. It is possible to use both color key sets for the same window or for two separate windows.

The two sets of color key ranges should not overlap and if they do the overlap colors are treated as if they are part of Color Key 0 and not part of Color Key 1.

Assuming that there is no overlap with other higher priority window with color key not matched, if a window color key is enabled for any overlap condition and the window pixel is not within the color key range (key not match), then the window pixel will not be blended with other overlapping window pixels but it will be weighted. The weight is not dependent on the overlap condition it is controlled by the same set of parameters for all overlapping condition.

Assuming that there is no overlap with other higher priority window with color key not matched, if a window color key is enabled for a particular overlap condition and the window pixel is within the color key range (key match), then the window pixel will be weighted and blended with other overlapping pixels and this is controlled separately for each overlap condition.

Assuming that there is no overlap with other higher priority window with color key not matched, if a window color key is disabled then the window pixel will be blended with other overlapping pixels and this is controlled separately for each overlap condition.

Display Color Key parameters

For B4G4R4A4, B5G6R5A, B5G6R5 mode, color key should be compared prior to color conversion to 24-bpp and unused least significant bits of the pixel are filled with zeros.

For palletized mode, color key is compared prior to color palette and the palletized color is compared against the green color key values/mask.

For YUV mode, U and V are offset by +128 before performing the color key comparison. In all cases, color key is compared prior to horizontal/vertical scaling filter and prior to digital vibrance control.

Both upper and lower values are inclusive.

Offset: 70fh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	A_BLEND_WEIGHT1_NOKEY: Window blend weight 1 for color key not match areas. This is used only for alpha weight with 1-bit alpha and alpha value of 1.
15:8	X	A_BLEND_WEIGHT0_NOKEY: Window blend weight 0 for color key not match areas. For alpha weight, this is used for 1-bit alpha with value of 0.
0	X	A_BLEND_CONTROL_NOKEY: Window blend control for color key not match areas. 0 = Fix weight using window blend weight 0 for color key not matched. 1 = Alpha weight using the alpha value. This is only valid if the window color format includes an alpha value. For 1-bit alpha value, if the alpha is 0 window blend weight 0 is used and if alpha is 1, window blend weight 1 is used. 0 = FIX_WEIGHT

Bit	Reset	Description
		1 = ALPHA_WEIGHT

29.10.18 DC_WIN_A_BLEND_1WIN_0

Blend Control for this window area where it does not overlap with other windows.

Offset: 710h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	A_BLEND_WEIGHT1_1WIN: Window blend weight 1 for color key disabled or color key enabled with key matched areas. This is used only for alpha weight with 1-bit alpha and alpha value of 1.
15:8	X	A_BLEND_WEIGHT0_1WIN: Window blend weight 0 for color key disabled or color key enabled with key matched areas. For alpha weight, this is used for 1-bit alpha with value of 0.
2	X	A_BLEND_CONTROL_1WIN: Window blend control in area where it does not overlap with other windows and either color key disabled or color key enabled with key matched. 0 = Fix weight using the window blend weight 0 if color key disabled or color key 0 enabled with key matched, or using the window blend weight 1 if color key 1 enabled with key matched. 1 = Alpha weight using the alpha value. This is only valid if the window color format includes an alpha value. For 1-bit alpha value, if the alpha is 0 window blend weight 0 is used and if alpha is 1, window blend weight 1 is used. 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT
1:0	X	A_CKEY_ENABLE_1WIN: Window color key enable 0 = Color Key 0 and 1 Disable 1 = Color Key 0 Enable 2 = Color Key 1 Enable 3 = Color Key 0 and 1 Enable 0 = NOKEY 1 = CKEY0 2 = CKEY1 3 = CKEY01

29.10.19 DC_WIN_A_BLEND_2WIN_B_0

Blend Control for this window area that overlaps with window B only.

Offset: 711h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	A_BLEND_WEIGHT1_2WIN_B: Window blend weight 1 for color key disabled or color key enabled with key matched areas. This is used only for alpha weight with 1-bit alpha and alpha value of 1.
15:8	X	A_BLEND_WEIGHT0_2WIN_B: Window blend weight 0 for color key disabled or color key enabled with key matched areas. For alpha weight, this is used for 1-bit alpha with value of 0.
3:2	X	A_BLEND_CONTROL_2WIN_B: Window blend control in area where either color key disabled or color key enabled with key matched. 0 = Fix weight using the window blend weight 0 if color key disabled or color key 0 enabled with key matched, or using the window blend weight 1 if color key 1 enabled with key matched. 1 = Alpha weight using the alpha value. This is only valid if the window color format includes an alpha value. For 1-bit alpha value, if the alpha is 0 window blend weight 0 is used and if alpha is 1, window blend weight 1 is used. 2 = Dependent weight, in this case, the window blend weight is 1 - the weights of the other window that overlaps with this window. Only 1 of the overlapping windows may have this setting. 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT 2 = DEPENDENT_WEIGHT

Bit	Reset	Description
1:0	X	A_CKEY_ENABLE_2WIN_B: Window color key enable 0 = Color Key 0 and 1 Disable 1 = Color Key 0 Enable 2 = Color Key 1 Enable 3 = Color Key 0 and 1 Enable 0 = NOKEY 1 = CKEY0 2 = CKEY1 3 = CKEY01

29.10.20 DC_WIN_A_BLEND_2WIN_C_0

Blend Control for this window area that overlaps with window C only.

Offset: 712h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	A_BLEND_WEIGHT1_2WIN_C: Window blend weight 1 for color key disabled or color key enabled with key matched areas. This is used only for alpha weight with 1-bit alpha and alpha value of 1.
15:8	X	A_BLEND_WEIGHT0_2WIN_C: Window blend weight 0 for color key disabled or color key enabled with key matched areas. For alpha weight, this is used for 1-bit alpha with value of 0.
3:2	X	A_BLEND_CONTROL_2WIN_C: Window blend control in area where either color key disabled or color key enabled with key matched. 0 = Fix weight using the window blend weight 0 if color key disabled or color key 0 enabled with key matched, or using the window blend weight 1 if color key 1 enabled with key matched. 1 = Alpha weight using the alpha value. Only valid if the window color format includes an alpha value. For 1-bit alpha value, if the alpha is 0 window blend weight 0 is used and if alpha is 1, window blend weight 1 is used. 2 = Dependent weight, in this case, the window blend weight is 1 - the weights of the other window that overlaps with this window. Only 1 of the overlapping windows may have this setting. 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT 2 = DEPENDENT_WEIGHT
1:0	X	A_CKEY_ENABLE_2WIN_C: Window color key enable 0 = Color Key 0 and 1 Disable 1 = Color Key 0 Enable 2 = Color Key 1 Enable 3 = Color Key 0 and 1 Enable 0 = NOKEY 1 = CKEY0 2 = CKEY1 3 = CKEY01

29.10.21 DC_WIN_A_BLEND_3WIN_BC_0

Blend Control for this window area that overlaps with windows B and C only.

Offset: 713h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	A_BLEND_WEIGHT1_3WIN_BC: Window blend weight 1 for color key disabled or color key enabled with key matched areas. This is used only for alpha weight with 1-bit alpha and alpha value of 1.
15:8	X	A_BLEND_WEIGHT0_3WIN_BC: Window blend weight 0 for color key disabled or color key enabled with key matched areas. For alpha weight, this is used for 1-bit alpha with value of 0.
3:2	X	A_BLEND_CONTROL_3WIN_BC: Window blend control in area where either color key disabled or color key enabled with key matched. 0 = Fix weight using the window blend weight 0 if color key disabled or color key 0 enabled with key matched, or using the window blend weight 1 if color key 1 enabled with key matched. 1 = Alpha weight using the alpha value. This is only valid if the window color format includes an alpha value. For 1-bit alpha value, if the alpha is 0 window blend weight 0 is used and if alpha is 1, window blend weight

		1 is used. 2 = Dependent weight, in this case, the window blend weight is 1 - the weights of the other window that overlaps with this window. Only 1 of the overlapping windows may have this setting. 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT 2 = DEPENDENT_WEIGHT
1:0	X	A_CKEY_ENABLE_3WIN_BC: Window color key enable 0 = Color Key 0 and 1 Disable 1 = Color Key 0 Enable 2 = Color Key 1 Enable 3 = Color Key 0 and 1 Enable 0 = NOKEY 1 = CKEY0 2 = CKEY1 3 = CKEY01

29.10.22 DC_WIN_A_HP_FETCH_CONTROL_0

Window A High Priority Avoidance Fetch Parameters

This register gives extra information to the Memory Controller Client Interface (MCCIF) about the number of memory words that will be fetched per scan line and about the rate at which those words are consumed. This allows the MCCIF to more accurately arbitrate memory accesses to prevent the use of the High Priority signal. Use of this signal causes all other client accesses to be blocked in preference to the client asserting HP.

This is a state which is sometime necessary for Display as it is an isochronous client and MUST be serviced in a timely manner. However, use of this signal should be avoided if possible. These parameters help the MCCIF avoid the over-use of HP.

Offset: 714h | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	0x0	A_FETCH_INFO_ENABLE: Enables the sending of the Window A fetch information. For compatibility with earlier devices, this defaults to DISABLE. 0 = DISABLE : This bit should be enabled only for 12-bpp 1 = ENABLE
30:16	X	A_WORDS_PER_LINE: Window A memory fetch words per scan line. This value is in memory fetch words: Multiples of 16 bytes for Tegra 2 Processor Series devices. It is computed as follows: A_WORDS_PER_LINE = (A_SIZE.A_H_SIZE * (bytes per pixel) + 15) >> 4 bytes per pixel is determined by the pixel format.
15:0	X	A_CYCLES_PER_WORD: Window A clock cycles per memory fetch word. The value of this field is essentially a measure of the data consumption rate for window A. It is computed as follows: A_CYCLES_PER_WORD = A_DDA_INCREMENT.A_H_DDA_INCREMENT / (bytes per pixel) Note that the format for this value is a fixed-point fractional value with 8 bits of integer precision and 8 bits of fractional precision. In other words, it is an '8.8' number. For example, if there is no scaling of the input image, the DDA increment will be 4096. With 32-bit RGBA pixels there will be 4 bytes per pixel, so CYCLES_PER_WORD will be ... 4096 / 4 = 1024, or 4.0 expressed in the 8.8 format. Any scaling performed on the pixels will change the rate at which pixels are consumed. Scaling up will increase the value of DDA increment and will therefore increase the number of cycles between memory fetches. Conversely, scaling down will decrease the value of DDA increment and memory fetches will occur more frequently.

29.11 WINBUF_A Registers

The registers under DC_WINBUF are triple-buffered.

29.11.1 DC_WINBUF_A_START_ADDR_0

Window A Start Address

Overview

START_ADDR, BUF_STRIDE, LINE_STRIDE, ADDR_H_OFFSET, ADDR_V_OFFSET combined, specify the starting address of a window buffer in the memory surface. Generally START_ADDR is programmed with the starting address of the memory surface. H/V_OFFSET specify the address offsets of the pixel at the beginning of the window, with respect to the starting address of the memory surface. (There is an exception to that when memory surface is not well aligned, see 3-ii below.)

Note that "beginning of the window" here means the top-left corner of a window in normal mode, top-right corner in horizontal flipping, bottom-left corner in vertical flipping, and bottom-right in horizontal+vertical flipping.

Starting address calculation

These formulae are same for both tiled or linear mode. However, for linear mode, the addresses calculated are real physical addresses, but for tile mode, these addresses will be translated to the real physical addresses by the memory controller client before used.

- When a window is host triggered, starting address of a window is calculated as follows by HW.
  - non-yuv-planar modes:

    starting-address = START_ADDR + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET
  - yuv-planar modes:

    y-starting-address = START_ADDR + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET

    u-starting-address = START_ADDR_U + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2

    v-starting-address = START_ADDR_V + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2 where denom1/denom2 equal to 1 or 2 depending on the actual planar format, derived natively by HW.
- When a window is non-host triggered, starting address of a window buffer is calculated as below.
  - non-yuv-planar mode:

    starting-address = START_ADDR + BUF_STRIDE * buf_index + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET
  - yuv-planar mode:

    y-starting-address = START_ADDR + BUF_STRIDE * buf_index + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET

    u-starting-address = START_ADDR_U + UV_BUF_STRIDE * buf_index + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2

    v-starting-address = START_ADDR_V + UV_BUF_STRIDE * buf_index + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2 where denom1/denom2 equal to 1 or 2 depending on the actual planar format, derived natively by HW.

    buf_index is the index transmitted by the triggering module pointing to the first buffer of a frame.

Programming Restrictions

i. For tiled address mode:

Image surface can only aligned to multiples of 256, thus the following restrictions.

- START_ADDR, START_ADDR_U, START_ADDR_V need to be multiples of 256.

- BUF_STRIDE, UV_BUF_STRIDE need to be multiples of 256

- LINE_STRIDE, UV_LINE_STRIDE need to be multiples of 16

- ADDR_H_OFFSET needs to be multiple of 2 in yuv planar format (the last bit is ignored), but with no restrictions on other color formats.

- ADDR_V_OFFSET has no restrictions

ii. For linear address mode:

Image surface can be aligned 2, 4 or 8 bytes, depending on the color formats.

As an additional restriction for display, START_ADDR, START_ADDR_U and START_ADDR_V need to be multiples of 16.

When a surface is not aligned to 16 bytes, program START_ADDR with the memory surface address with its least 4 significant bits zeroed out and add these 4 address bits to the original H_OFFSET. (So the formula in 2-i,ii still hold)

- For all formats:

-- START_ADDR, START_ADDR_U and START_ADDR_V need to be multiples of 16.

- For 16-bpp formats,

-- (START_ADDR+H_OFFSET) need to be multiple of 2. The least significant bit of H_OFFSTE is ignored.

- For 32-bpp formats,

-- (START_ADDR+H_OFFSTE) needs to be multiple of 4. The least two significant bits of H_OFFSTE are ignored.

- For yuv planar formats:

-- BUF_STRIDE, UV_BUF_STRIDE:

BUF_STRIDE[2:1]=UV_BUF_STRIDE[1:0]

or as a stricter constraint: BUF_STRIDE be multiple of 8, UV_BUF_STRIDE be multiple of 4.

-- LINE_STRIDE, UV_LINE_STRIDE:

LINE_STRIDE and UV_LINE_STRIDE need to be at least 16.

LINE_STRIDE needs to be multiple of 8, UV_LINE_STRIDE needs to be multiple of 4.

-- ADDR_H_OFFSET: Needs to be multiple of 2. If needs to point to odd pixel position, program ADDR_H_OFFSET to be the previous position (or the next position if H-flipped) and program H_INITIAL_DDA bit 12 to 1.

-- ADDR_V_OFFSET: Needs to be multiple of 2. If needs to point to odd line number, program ADDR_V_OFFSTE to be the previous line number (or next line number if V-flipped) and program V_INITIAL_DDA bit 12 to 1.

iii. Memory allocation for non-host triggered case:

When a window buffer is not controlled by host (software) then a frame may be stored in multiple buffers. In this case, the buffers must be contiguous in the memory because display will use the same luma or chroma line strides for all lines in the frame. Also buffer wraparound must not occur in the middle of the displayed part of the frame.

The controlling module will send frame start and frame end indicators (flags) to display module to indicate the beginning and end of frame. Buffer start address is latched when frame start flag is active but the actual buffer start address is not switched

until frame end flag is active. In the case where one buffer correspond to one frame then frame start and frame end flag are active every time a buffer index is sent.

Offset: 800h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	A_START_ADDR: Window A Start Address This is a byte address. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSB are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies start address for the Y plane.

29.11.2 DC_WINBUF_A_START_ADDR_NS_0

Window A Shadowed Start Address

Offset: 801h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	A_START_ADDR_NS: Window A Shadowed Start Address This is ARM set shadow of Start Address.

29.11.3 DC_WINBUF_A_ADDR_H_OFFSET_0

Window A Horizontal address offset

Offset: 806h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	A_ADDR_H_OFFSET: Window A Horizontal address offset This is a byte address. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSB are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies horizontal offset of Y plane. The horizontal offsets of U/V plane is derived by HW.

29.11.4 DC_WINBUF_A_ADDR_H_OFFSET_NS_0

Window A Shadowed Horizontal address offset

Offset: 807h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	A_ADDR_H_OFFSET_NS: Window A Shadowed Horizontal address offset This is ARM set shadow of ADDR_H_OFFSET

29.11.5 DC_WINBUF_A_ADDR_V_OFFSET_0

Window A Vertical address offset

Offset: 808h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	A_ADDR_V_OFFSET: Window A Vertical address offset This is a byte address. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSB are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies vertical offset of Y plane. The vertical offsets of U/V plane is derived by HW.

29.11.6 DC_WINBUF_A_ADDR_V_OFFSET_NS_0

Window A Shadowed Vertical address offset

Offset: 809h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	A_ADDR_V_OFFSET_NS: Window A Shadowed Vertical address offset This is ARM set shadow of ADDR_V_OFFSET

29.11.7 DC_WINBUF_A_UFLOW_STATUS

Window A FIFO Underflow Status Register

Offset: 80ah | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
30	X	COUNT_OFLOW: Flag bit that indicates that the underflow event counter has overflowed. There were too many events. If COUNT_OFLOW is set, UFLOW_COUNT is meaningless. Cleared on write.
23:0	X	UFLOW_COUNT: Underflow count. This field indicates the number of contiguous groups of output pixels for which there was no data in the FIFO. For example, if the valid from the FIFO is low for 10 consecutive cycles and then goes high, the counter will increment by one. Reset to zero on write.

29.12 Window B (WINC_B) Registers

These registers control window B parameters

29.12.1 DC_WINC_B_COLOR_PALETTE_0

Window B Color Palette

This is used for palletized data format (color depth of 8-bpp or less) or for gamma correction for non-palletized data formats (color depth of more than 8-bpp).

Each window has its own color palette which consists of three 256x8 register file which can be written by host and indexed (read) by the window.

For palletized data format less than 8-bpp the pixel data is aligned to least significant bits of the palette index (address) and the remaining upper bits are filled with the corresponding bits of the Palette Color Extension. For example, for 4-bpp mode, the pixel data occupies bits 3-0 of the palette index and bits 7-4 of the palette index are set to bits 7-4 of the Palette Color Extension.

Note that host read is assumed to be not needed - software can cache the color palette in system memory.

This is an array of 256 identical register entries; the register fields below apply to each entry.

Offset: 500h..5ffh | Read/Write: WO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	B_COLOR_PALETTE_B: Blue Color Palette
15:8	X	B_COLOR_PALETTE_G: Green Color Palette

Bit	Reset	Description
7:0	X	B_COLOR_PALETTE_R: Red Color Palette

29.12.2 DC_WINC_B_PALETTE_COLOR_EXT_0

Window B Palette Color Extension

Palette extension for 1-bpp, 2-bpp, and 4-bpp. These bits provide the upper most significant bits for indexing the color palette. Supported for window A only.

XXX should ifdef for window A, but currently window B spec is assumed to be a superset.

Offset: 600h | Read/Write: R/W | Reset: 0bxxxxxxx

Bit	Reset	Description
7:1	X	B_PALETTE_COLOR_EXT: Window B Palette Color Extension bits 7-1 are used for 1-bpp mode bits 7-2 are used for 2-bpp mode bits 7-4 are used for 4-bpp mode

29.12.3 DC_WINC_B_H_FILTER_P00_0

Window B Horizontal Filter phase 00

Horizontal scaling filter coefficients

Horizontal scaling filter is a 6-tap filter with 4-bit positional phase.

- Coefficients 0 and 5 are 3-bit signed value ranging from -4 to 3.
- Coefficients 1 and 4 are 5-bit signed value ranging from -16 to 15.
- Coefficients 2 and 3 are 8-bit unsigned value ranging from 0 to 128.
- Coefficient 0 is the multiplier for the earliest pixel (P0) in the group of 6-pixel and coefficient 5 is the multiplier for the latest pixel (P5) in the group. The output pixel positional phase is defined as centered in P2 if the positional phase is 0 or proportionally in between P2 and P3 if the positional phase is larger than 0.

Sum of all coefficients for each phase should be 128 typically and software should never program the the sum of all coefficients for a phase to be more than 128. For each horizontal positional phase, the 6 filter coefficients requires 32 reg bits. Note that color value ranges from 0 to 255 for Y, R, G, B and -128 to 127 for U and V.

Offset: 601h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	B_H_FILTER_P00C5: Phase 00 coefficient 5 (typically 0)
28:24	X	B_H_FILTER_P00C4: Phase 00 coefficient 4 (typically 0)
23:16	X	B_H_FILTER_P00C3: Phase 00 coefficient 3 (typically 0)
15:8	X	B_H_FILTER_P00C2: Phase 00 coefficient 2 (typically 128)
7:3	X	B_H_FILTER_P00C1: Phase 00 coefficient 1 (typically 0)
2:0	X	B_H_FILTER_P00C0: Phase 00 coefficient 0 (typically 0)

29.12.4 DC_WINC_B_H_FILTER_P01_0

Window B Horizontal Filter phase 01

Offset: 602h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	B_H_FILTER_P01C5: Phase 01 coefficient 5 (typically 1)
28:24	X	B_H_FILTER_P01C4: Phase 01 coefficient 4 (typically -2)
23:16	X	B_H_FILTER_P01C3: Phase 01 coefficient 3 (typically 8)
15:8	X	B_H_FILTER_P01C2: Phase 01 coefficient 2 (typically 124)
7:3	X	B_H_FILTER_P01C1: Phase 01 coefficient 1 (typically -4)
2:0	X	B_H_FILTER_P01C0: Phase 01 coefficient 0 (typically 1)

29.12.5 DC_WINC_B_H_FILTER_P02_0

Window B Horizontal Filter phase 02

Offset: 603h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	B_H_FILTER_P02C5: Phase 02 coefficient 5 (typically 1)
28:24	X	B_H_FILTER_P02C4: Phase 02 coefficient 4 (typically -5)
23:16	X	B_H_FILTER_P02C3: Phase 02 coefficient 3 (typically 17)
15:8	X	B_H_FILTER_P02C2: Phase 02 coefficient 2 (typically 122)
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)

29.12.6 DC_WINC_B_H_FILTER_P03_0

Window B Horizontal Filter phase 03

Offset: 604h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	B_H_FILTER_P03C5: Phase 03 coefficient 5 (typically 2)
28:24	X	B_H_FILTER_P03C4: Phase 03 coefficient 4 (typically -7)
23:16	X	B_H_FILTER_P03C3: Phase 03 coefficient 3 (typically 27)
15:8	X	B_H_FILTER_P03C2: Phase 03 coefficient 2 (typically 115)
7:3	X	B_H_FILTER_P03C1: Phase 03 coefficient 1 (typically -11)
2:0	X	B_H_FILTER_P03C0: Phase 03 coefficient 0 (typically 2)

29.12.7 DC_WINC_B_H_FILTER_P04_0

Window B Horizontal Filter phase 04

Offset: 605h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	B_H_FILTER_P04C5: Phase 04 coefficient 5 (typically 2)
28:24	X	B_H_FILTER_P04C4: Phase 04 coefficient 4 (typically -9)
23:16	X	B_H_FILTER_P04C3: Phase 04 coefficient 3 (typically 37)
15:8	X	B_H_FILTER_P04C2: Phase 04 coefficient 2 (typically 109)
7:3	X	B_H_FILTER_P04C1: Phase 04 coefficient 1 (typically -13)
2:0	X	B_H_FILTER_P04C0: Phase 04 coefficient 0 (typically 2)

29.12.8 DC_WINC_B_H_FILTER_P05_0

Window B Horizontal Filter phase 05

Offset: 606h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	B_H_FILTER_P05C5: Phase 05 coefficient 5 (typically 2)
28:24	X	B_H_FILTER_P05C4: Phase 05 coefficient 4 (typically -11)
23:16	X	B_H_FILTER_P05C3: Phase 05 coefficient 3 (typically 47)
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)

29.12.9 DC_WINC_B_H_FILTER_P06_0

Window B Horizontal Filter phase 06

Offset: 607h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	B_H_FILTER_P06C5: Phase 06 coefficient 5 (typically 3)
28:24	X	B_H_FILTER_P06C4: Phase 06 coefficient 4 (typically -13)
23:16	X	B_H_FILTER_P06C3: Phase 06 coefficient 3 (typically 56)
15:8	X	B_H_FILTER_P06C2: Phase 06 coefficient 2 (typically 94)
7:3	X	B_H_FILTER_P06C1: Phase 06 coefficient 1 (typically -15)
2:0	X	B_H_FILTER_P06C0: Phase 06 coefficient 0 (typically 3)

29.12.10 DC_WINC_B_H_FILTER_P07_0

Window B Horizontal Filter phase 07

Offset: 608h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	B_H_FILTER_P07C5: Phase 07 coefficient 5 (typically 3)
28:24	X	B_H_FILTER_P07C4: Phase 07 coefficient 4 (typically -14)
23:16	X	B_H_FILTER_P07C3: Phase 07 coefficient 3 (typically 67)
15:8	X	B_H_FILTER_P07C2: Phase 07 coefficient 2 (typically 85)
7:3	X	B_H_FILTER_P07C1: Phase 07 coefficient 1 (typically -16)
2:0	X	B_H_FILTER_P07C0: Phase 07 coefficient 0 (typically 3)

29.12.11 DC_WINC_B_H_FILTER_P08_0

Window B Horizontal Filter phase 08

Offset: 609h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	B_H_FILTER_P08C5: Phase 08 coefficient 5 (typically 3)
28:24	X	B_H_FILTER_P08C4: Phase 08 coefficient 4 (typically -15)
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)

29.12.12 DC_WINC_B_H_FILTER_P09_0

Window B Horizontal Filter phase 09

Offset: 60ah | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	B_H_FILTER_P09C5: Phase 09 coefficient 5 (typically 3)
28:24	X	B_H_FILTER_P09C4: Phase 09 coefficient 4 (typically -16)
23:16	X	B_H_FILTER_P09C3: Phase 09 coefficient 3 (typically 85)
15:8	X	B_H_FILTER_P09C2: Phase 09 coefficient 2 (typically 67)
7:3	X	B_H_FILTER_P09C1: Phase 09 coefficient 1 (typically -14)
2:0	X	B_H_FILTER_P09C0: Phase 09 coefficient 0 (typically 3)

29.12.13 DC_WINC_B_H_FILTER_P0A_0

Window B Horizontal Filter phase 0A

Offset: 60bh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	B_H_FILTER_P0AC5: Phase 0A coefficient 5 (typically 3)
28:24	X	B_H_FILTER_P0AC4: Phase 0A coefficient 4 (typically -15)
23:16	X	B_H_FILTER_P0AC3: Phase 0A coefficient 3 (typically 94)
15:8	X	B_H_FILTER_P0AC2: Phase 0A coefficient 2 (typically 56)
7:3	X	B_H_FILTER_P0AC1: Phase 0A coefficient 1 (typically -13)
2:0	X	B_H_FILTER_P0AC0: Phase 0A coefficient 0 (typically 3)

29.12.14 DC_WINC_B_H_FILTER_P0B_0

Window B Horizontal Filter phase 0B

Offset: 60ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	B_H_FILTER_P0BC5: Phase 0B coefficient 5 (typically 3)
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)

29.12.15 DC_WINC_B_H_FILTER_P0C_0

Window B Horizontal Filter phase 0C

Offset: 60dh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	B_H_FILTER_P0CC5: Phase 0C coefficient 5 (typically 2)
28:24	X	B_H_FILTER_P0CC4: Phase 0C coefficient 4 (typically -13)
23:16	X	B_H_FILTER_P0CC3: Phase 0C coefficient 3 (typically 109)
15:8	X	B_H_FILTER_P0CC2: Phase 0C coefficient 2 (typically 37)
7:3	X	B_H_FILTER_P0CC1: Phase 0C coefficient 1 (typically -9)
2:0	X	B_H_FILTER_P0CC0: Phase 0C coefficient 0 (typically 2)

29.12.16 DC_WINC_B_H_FILTER_P0D_0

Window B Horizontal Filter phase 0D

Offset: 60eh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	B_H_FILTER_P0DC5: Phase 0D coefficient 5 (typically 2)
28:24	X	B_H_FILTER_P0DC4: Phase 0D coefficient 4 (typically -11)
23:16	X	B_H_FILTER_P0DC3: Phase 0D coefficient 3 (typically 115)
15:8	X	B_H_FILTER_P0DC2: Phase 0D coefficient 2 (typically 27)
7:3	X	B_H_FILTER_P0DC1: Phase 0D coefficient 1 (typically -7)
2:0	X	B_H_FILTER_P0DC0: Phase 0D coefficient 0 (typically 2)

29.12.17 DC_WINC_B_H_FILTER_P0E_0

Window B Horizontal Filter phase 0E

Offset: 60fh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	B_H_FILTER_P0EC5: Phase 0E coefficient 5 (typically 1)
28:24	X	B_H_FILTER_P0EC4: Phase 0E coefficient 4 (typically -8)
23:16	X	B_H_FILTER_P0EC3: Phase 0E coefficient 3 (typically 122)
15:8	X	B_H_FILTER_P0EC2: Phase 0E coefficient 2 (typically 17)
7:3	X	B_H_FILTER_P0EC1: Phase 0E coefficient 1 (typically -5)
2:0	X	B_H_FILTER_P0EC0: Phase 0E coefficient 0 (typically 1)

29.12.18 DC_WINC_B_H_FILTER_P0F_0

Window B Horizontal Filter phase 0F

Offset: 610h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	B_H_FILTER_P0FC5: Phase 0F coefficient 5 (typically 1)
28:24	X	B_H_FILTER_P0FC4: Phase 0F coefficient 4 (typically -4)
23:16	X	B_H_FILTER_P0FC3: Phase 0F coefficient 3 (typically 124)
15:8	X	B_H_FILTER_P0FC2: Phase 0F coefficient 2 (typically 8)
7:3	X	B_H_FILTER_P0FC1: Phase 0F coefficient 1 (typically -2)
2:0	X	B_H_FILTER_P0FC0: Phase 0F coefficient 0 (typically 1)

29.12.19 DC_WINC_B_CSC_YOF_0

Window B CSC Y Offset

Color Space Conversion coefficients.

The CSC can be used for YUV to RGB conversion with brightness and hue/saturation control.

The CSC can only be enabled for window B controlled by CSC_ENABLE register bits.

For Y color, the Y offset is applied first and saturation (clipping) is performed immediately after the Y offset is applied.

R = sat(KYRGB * sat(Y + YOF) + KUR * U + KVR * V)

G = sat(KYRGB * sat(Y + YOF) + KUG * U + KVG * V)

B = sat(KYRGB * sat(Y + YOF) + KUB * U + KVB * V)

Saturation and rounding is performed in the range of 0 to 255 for the above equations.

Typical values are:

YOF = -16.000, KYRGB = 1.1644

KUR = 0.0000, KVR = 1.5960

KUG = -0.3918, KVG = -0.8130

KUB = 2.0172, KVB = 0.0000

KUR and KVB are typically 0.0000 but they may be programmed non-zero for hue rotation.

The CSC can also take RGB input, in which case YOF, KVB, KUG, KUR should be programmed to 0 and KYRGB will be forced to 0 by the hardware for generating R and B. KYRGB will not be forced to 0 for generating G. KVR, KYRGB, and KUB can be programmed to 1.0 or used as gain control for R, G, B correspondingly.

Note that color value ranges from 0 to 255 for Y, R, G, B and -128 to 127 for U and V.

Offset: 611h | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	B_CSC_YOF: Y Offset in s.7.0 format

29.12.20 DC_WINC_B_CSC_KYRGB_0

Window B CSC Y Coefficient (gain) for RGB

Offset: 612h | Read/Write: R/W | Reset: 0bxxxxxxxxxx

Bit	Reset	Description
9:0	X	B_CSC_KYRGB: Y Gain for R, G, B colors in 2.8 format

29.12.21 DC_WINC_B_CSC_KUR_0

Window B CSC U coefficient for R

Offset: 613h | Read/Write: R/W | Reset: 0bxxxxxxxxxxx

Bit	Reset	Description
10:0	X	B_CSC_KUR: U coefficients for R in s.2.8 format

29.12.22 DC_WINC_B_CSC_KVR_0

Window B CSC V coefficient for R

Offset: 614h | Read/Write: R/W | Reset: 0bxxxxxxxxxxx

Bit	Reset	Description
10:0	X	B_CSC_KVR: V coefficients for R in s.2.8 format

29.12.23 DC_WINC_B_CSC_KUG_0

Window B CSC U coefficient for G

Offset: 615h | Read/Write: R/W | Reset: 0bxxxxxxxxxx

Bit	Reset	Description
9:0	X	B_CSC_KUG: U coefficients for G in s.1.8 format

29.12.24 DC_WINC_B_CSC_KVG_0

Window B CSC V coefficient for G

Offset: 616h | Read/Write: R/W | Reset: 0bxxxxxxxxxx

Bit	Reset	Description
9:0	X	B_CSC_KVG: V coefficients for G in s.1.8 format

29.12.25 DC_WINC_B_CSC_KUB_0

Window B CSC U coefficient for B

Offset: 617h | Read/Write: R/W | Reset: 0bxxxxxxxxxxx

Bit	Reset	Description
10:0	X	B_CSC_KUB: U coefficients for B in s.2.8 format

29.12.26 DC_WINC_B_CSC_KVB_0

Window B CSC V coefficient for B

Offset: 618h | Read/Write: R/W | Reset: 0bxxxxxxxxxxx

Bit	Reset	Description
10:0	X	B_CSC_KVB: V coefficients for B in s.2.8 format

29.12.27 DC_WINC_B_V_FILTER_P00_0

Window B Vertical Filter phase 00

Vertical scaling filter coefficients

Vertical scaling filter is a 2-tap filter with 4-bit positional phase.

Coefficients 0 and 1 are 8-bit unsigned value ranging from 0 to 128.

Coefficient 0 is the multiplier for the earlier pixel (P0) in the group of 2-pixel and coefficient 1 is the multiplier for the later pixel (P1) in the group. The output pixel positional phase is defined as centered in P0 if the positional phase is 0 or proportionally in between P0 and P1 if the positional phase is larger than 0.

Sum of all coefficients for each phase should be 128 typically therefore coefficient 1 can be calculated from (1 - coefficient 0) and therefore only coefficient 0 is programmed.

For each vertical positional phase, the filter coefficient requires 8 reg bits. Note that color value ranges from 0 to 255 for Y, R, G, B and -128 to 127 for U and V.

Offset: 619h | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	B_V_FILTER_P00C0: Phase 00 coefficient 0 (typically 128)

29.12.28 DC_WINC_B_V_FILTER_P01_0

Window B Vertical Filter phase 01

Offset: 61ah | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	B_V_FILTER_P01C0: Phase 01 coefficient 0 (typically 120)

29.12.29 DC_WINC_B_V_FILTER_P02_0

Window B Vertical Filter phase 02

Offset: 61bh | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	B_V_FILTER_P02C0: Phase 02 coefficient 0 (typically 112)

29.12.30 DC_WINC_B_V_FILTER_P03_0

Window B Vertical Filter phase 03

Offset: 61ch | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	B_V_FILTER_P03C0: Phase 03 coefficient 0 (typically 104)

29.12.31 DC_WINC_B_V_FILTER_P04_0

Window B Vertical Filter phase 04

Offset: 61dh | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	B_V_FILTER_P04C0: Phase 04 coefficient 0 (typically 96)

29.12.32 DC_WINC_B_V_FILTER_P05_0

Window B Vertical Filter phase 05

Offset: 61eh | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	B_V_FILTER_P05C0: Phase 05 coefficient 0 (typically 88)

29.12.33 DC_WINC_B_V_FILTER_P06_0

Window B Vertical Filter phase 06

Offset: 61fh | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	B_V_FILTER_P06C0: Phase 06 coefficient 0 (typically 80)

29.12.34 DC_WINC_B_V_FILTER_P07_0

Window B Vertical Filter phase 07

Offset: 620h | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	B_V_FILTER_P07C0: Phase 07 coefficient 0 (typically 72)

29.12.35 DC_WINC_B_V_FILTER_P08_0

Window B Vertical Filter phase 08

Offset: 621h | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	B_V_FILTER_P08C0: Phase 08 coefficient 0 (typically 64)

29.12.36 DC_WINC_B_V_FILTER_P09_0

Window B Vertical Filter phase 09

Offset: 622h | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	B_V_FILTER_P09C0: Phase 09 coefficient 0 (typically 56)

29.12.37 DC_WINC_B_V_FILTER_P0A_0

Window B Vertical Filter phase 0A

Offset: 623h | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	B_V_FILTER_P0AC0: Phase 0A coefficient 0 (typically 48)

29.12.38 DC_WINC_B_V_FILTER_P0B_0

Window B Vertical Filter phase 0B

Offset: 624h | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	B_V_FILTER_P0BC0: Phase 0B coefficient 0 (typically 40)

29.12.39 DC_WINC_B_V_FILTER_P0C_0

Window B Vertical Filter phase 0C

Offset: 625h | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	B_V_FILTER_P0CC0: Phase 0C coefficient 0 (typically 32)

29.12.40 DC_WINC_B_V_FILTER_P0D_0

Window B Vertical Filter phase 0D

Offset: 626h | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	B_V_FILTER_P0DC0: Phase 0D coefficient 0 (typically 24)

29.12.41 DC_WINC_B_V_FILTER_P0E_0

Window B Vertical Filter phase 0E\

Offset: 627h | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	B_V_FILTER_P0EC0: Phase 0E coefficient 0 (typically 16)

29.12.42 DC_WINC_B_V_FILTER_P0F_0

Window B Vertical Filter phase 0F

Offset: 628h | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	B_V_FILTER_P0FC0: Phase 0F coefficient 0 (typically 8)

The registers under DC_WIN are double buffered

29.12.43 DC_WIN_B_WIN_OPTIONS_0

Window B Options

Class: Display Window Settings

Display Window B parameters

Offset: 700h | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxx0x0xxxxxxxxxxxxxx

Bit	Reset	Description
30	0x0	B_WIN_ENABLE: Window B Window enable 0 = DISABLE 1 = ENABLE
22	X	B_YUV_RANGE_EXPAND: Window B Enable range expansion in the cases where RANGEREDFRM is 1 from mpd. Formula: Y = clip((Y-128)*2 + 128); Cb = clip((Cb-128)*2 + 128); Cr = clip((Cr-128)*2 + 128); where clip() function clips between 0 and 255. 0= disable 1= enable 0 = DISABLE 1 = ENABLE
20	X	B_DV_ENABLE: Window B Digital Vibrance Enable 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
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. 0 = DISABLE 1 = ENABLE
8	X	B_H_FILTER_ENABLE: Window B H Filter Enable This controls H scaling filter and is effective only for non-palletized color modes. 0 = DISABLE 1 = ENABLE
6	X	B_COLOR_EXPAND: Window B 12/15/16/18-to-24 bpp color expansion This bit should be enabled only for 12-bpp B4G4R4A4, 15-bpp B5G5R5A, 16-bpp B5G6R5, 18-bpp B6G6R6 color modes. If enabled the color conversion is performed prior to horizontal scaling. 0 = DISABLE 1 = ENABLE
2	X	B_V_DIRECTION: Window B Vertical (Y) drawing Direction 0 = INCREMENT 1 = DECREMENT
0	X	B_H_DIRECTION: Window B Horizontal (X) drawing Direction 0 = INCREMENT 1 = DECREMENT

29.12.44 DC_WIN_B_BYTE_SWAP_0

Window B Byte Swap

Offset: 701h | Read/Write: R/W | Reset: 0bxx

Bit	Reset	Description
1:0	X	B_BYTE_SWAP: Window B Byte Swap This controls byte swap of frame data read from memory prior to any data processing in the display module. 00= no byte swap (3 2 1 0) 01= byte swap for each 2-byte word (2 3 0 1) 10= byte swap for each 4-byte word (0 1 2 3) 11= word swap for each 4-byte word (1 0 3 2) 0 = NOSWAP 1 = SWAP2 2 = SWAP4 3 = SWAP4HW

29.12.45 DC_WIN_B_BUFFER_CONTROL_0

Window B Buffer Control

Offset: 702h | Read/Write: R/W | Reset: 0b000

Bit	Reset	Description
2:0	0x0	B_BUFFER_CONTROL: Window B Buffer Control 0= Host (software) controlled 1= Video Input controlled 2= Encoder Pre-Processor controlled 3= MPEG Encoder controlled 4= StretchBLT or 2D other= reserved If window buffer selection is not controlled by host (software) then buffer start indexes are sent by the respective module specified by this parameter, and in this case, the buffer start address registers are used to specify frame stride and buffer offset for the calculated start address. 0 = HOST 1 = VI 2 = EPP 4 = SB2D 3 = MPEGE

29.12.46 DC_WIN_B_COLOR_DEPTH_0

Window B Color Depth For YCbCr data format, Cb and Cr are 8-bit unsigned values. For YUV data format, U and V are 8-bit signed values. YCbCr422R is similar to YCbCr422P but the Cb and Cr are shared vertically.

YUV422R is similar to YUV422P but the U and V are shared vertically. YCbCr422RA is same as YCbCr422R in memory and YUV422RA is same as YUV422R in memory but while reading from memory, for YCbCr422RA and YUV422RA, chroma averaging is applied for each pixel pair so that they can be processed as YUV422 by the display pipeline.

For YCbCr422R and YUV422R, every other chroma pixels are not used (discarded) by the display pipeline. B6x2G6x2R6x2A8 is similar to B8G8R6A8 but with the 2 lsb zeroed out. R6x2G6x2B6x2A8 is similar to R8G8B6A8 but with the 2 lsb zeroed out.

Offset: 703h | Read/Write: R/W | Reset: 0bxxxxx

Bit	Reset	Description
4:0	X	B_COLOR_DEPTH: Window B Color Depth Supported color depths are: P8 = 8-bpp (palletized) B4G4R4A4 = 12-bpp B4G4R4 B5G5R5A = 15-bpp B5G5R5 AB5G5R5 = 15-bpp B5G5R5 B5G6R5 = 16-bpp B5G6R5 B8G8R8A8 = 32-bpp B8G8R8A8 R8G8B8A8 = 32-bpp R8G8B8A8 B6x2G6x2R6x2A8 = 32-bpp B6G6R6A8 R6x2G6x2B6x2A8 = 32-bpp R6G6B6A8 YCbCr422 = 16-bpp YCbCr422 packed YUV422 = 16-bpp YUV422 YCbCr420P = 16-bpp YCbCr420 planar YUV420P = 16-bpp YUV420 planar YCbCr422P = 16-bpp YCbCr422 planar YUV422P = 16-bpp YUV422 planar YCbCr422R = 16-bpp YCbCr422 rotated planar YUV422R = 16-bpp YUV422 rotated planar YCbCr422RA= 16-bpp YCbCr422 rotated planar with chroma averaging YUV422RA = 16-bpp YUV422 rotated planar with chroma averaging 0 = P1 1 = P2 2 = P4 3 = P8 4 = B4G4R4A4 5 = B5G5R5A 6 = B5G6R5 7 = AB5G5R5 12 = B8G8R8A8 13 = R8G8B8A8 14 = B6x2G6x2R6x2A8 15 = R6x2G6x2B6x2A8 16 = YCbCr422 17 = YUV422 18 = YCbCr420P 19 = YUV420P 20 = YCbCr422P 21 = YUV422P 22 = YCbCr422R 23 = YUV422R 24 = YCbCr422RA 25 = YUV422RA

29.12.47 DC_WIN_B_POSITION_0

Window B Position

This register defines H position and size of Window B after scaling (if there is any)

Offset: 704h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	B_V_POSITION: Window B V Position This is specified with respect to the top edge of active display area.
12:0	X	B_H_POSITION: Window B H Position This is specified with respect to the left edge of active display area.

29.12.48 DC_WIN_B_SIZE_0

Window B Size

This register defines V position and size of Window B after scaling (if there is any)

Note: programming on window size should guarantee the whole window is inside the active area, otherwise extra rows/columns will be fetched and discarded which affects performance.

Offset: 705h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	B_V_SIZE: Window B V Size (lines) This is the vertical size after scaling.
12:0	X	B_H_SIZE: Window B H Size (pixels) This is the horizontal size after scaling.

29.12.49 DC_WIN_B_PRESCALED_SIZE_0

Window B Pre-scaled Size

This register defines Window B pre-scaled size.

The H pre-scaled size is needed to determine how many bytes to fetch from memory per line and this parameter must be programmed exactly as needed taking into account the scaling factor. For planar YUV or YCbCr data formats, this parameter refer to the H pre-scaled of the Y plane.

The total number of lines to be fetched from memory is determined by post-scale V size but V pre-scaled size is needed to 'clamp' the last valid line if the vertical DDA is exactly or slightly beyond the specified V pre-scaled size.

Design Note: H pre-scaled size ideally should be in terms of pixel but then hardware needs to convert this precisely to bytes to determine the amount of data to request from memory.

This could be a risky calculation - maybe this should be made optional on whether we use internal hardware to calculate or left it to software to calculate.

Offset: 706h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	B_V_PRESCALED_SIZE: Window B V Pre-scaled Size (lines) In 420P/422R/422RA formats, it must be even.
14:0	X	B_H_PRESCALED_SIZE: Window B H Pre-scaled Size (bytes) In 420P and 422P formats, it must be even.

29.12.50 DC_WIN_B_H_INITIAL_DDA_0

Window B H Initial DDA

Design Note: the first pixel of pre-scaled image is always used to output the first pixel so essentially this is the same as forcing the H Initial DDA integer portion to 1 initially even though user typically programs this to 0. If it makes the implementation easier, it is possible to force software to program the Initial DDA integer portion to 1. Similarly with the V Initial DDA.

Offset: 707h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	B_H_INITIAL_DDA: Window B H Initial DDA (4.12) This is typically programmed to 0.0

29.12.51 DC_WIN_B_V_INITIAL_DDA_0

Window B V Initial DDA

Offset: 708h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	B_V_INITIAL_DDA: Window B V Initial DDA (4.12) This is typically programmed to 0.0 for

Bit	Reset	Description
		both non-interlaced and interlaced sources.

29.12.52 DC_WIN_B_DDA_INCREMENT_0

Window B DDA Increment

DDA increment is typically calculated by dividing (Pre-scaled size in pixels - 1) by (Post-scaled size in pixels - 1). The result should be rounded up and expressed as 4.12 format (4-bit integer and 12-bit fraction). For non-filtered image this value can be slightly larger so that it is not missing the last row/column. Reference programming values (H and V should be calculated separately depending on filter on/off and sizes):

- Filter on: min(round((prescaled_size_in_pixels - 1) * 0x1000 / (post_scaled_size_in_pixels - 1)), MAX)
- Filter off: min(round(prescaled_size_in_pixels * 0x1000 / (post_scaled_size_in_pixels - 1) - 0.5), MAX)

Where the value of MAX is as follows:

For V_DDA_INCREMENT: 15.0 (0xF000)

For H_DDA_INCREMETN: 4.0 (0x4000) for 2 Bytes/pix formats.

8.0 (0x8000) for 4 Bytes/pix formats.

They are theoretically the biggest values that guarantees not displaying beyond an image boundary.

If the DDA increment is less than 1.0 then image will be up-scaled and if DDA increment is more than 1.0 then image will be down-scaled.

Offset: 709h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	B_V_DDA_INCREMENT: Window B Vertical DDA Increment (4.12) This should be set to 1.0 if there is no scaling. Maximum value is 15.0 regardless of the number of bytes per pixel.
15:0	X	B_H_DDA_INCREMENT: Window B Horizontal DDA Increment (4.12) This should be set to 1.0 if there is no scaling. The maximum value for downscaling depends on the number of bytes per pixel. For 4-byte/pixel modes (32-bpp) the maximum value is 4.0 and for all other modes the maximum value is 8.0.

29.12.53 DC_WIN_B_LINE_STRIDE_0

Window B Line Stride

Offset: 70ah | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	B_UV_LINE_STRIDE: Window B Line Stride for Chroma This is stride (in bytes) for planar YUV or YCbCr data formats for the chroma plane, with the restriction that it must be programmed to be multiples of 4 (16 if tiled or in horizontal flipping) This is not used (ignored) for other non-planar data formats.
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.

29.12.54 DC_WIN_B_BUF_STRIDE_0

Window B Buffer stride

Offset: 70bh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	B_BUF_STRIDE: Window B Buffer stride Buffer stride is used to calculate the buffer addresses when the window is triggered by non-host modules. Refer to the comment of of START_ADDR for programming guide. For YUV planar pixel format, this specifies buffer stride for the Y plane. The value is in bytes.

29.12.55 DC_WIN_B_UV_BUF_STRIDE_0

Window B Buffer stride for U/V plane

Offset: 70ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	B_UV_BUF_STRIDE: Window B This value is in bytes. For YUV planar pixel format, this specifies buffer stride for the U/V plane.

29.12.56 DC_WIN_B_BUFFER_ADDR_MODE_0

Memory Controller Tiling definitions

Offset: 70dh | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxx0

Bit	Reset	Description
16	0x0	B_UV_TILE_MODE: Window B Memory surface tiling mode For YUV planar pixel format, this specifies tiling mode for the U/V plane. 0 = LINEAR 1 = TILED
0	0x0	B_TILE_MODE: Window B Memory surface tiling mode For YUV planar pixel format, this specifies tiling mode for the Y plane. 0 = LINEAR 1 = TILED

29.12.57 DC_WIN_B_DV_CONTROL_0

Window B Digital Vibrance Control

If enabled, Digital Vibrance is applied after H and V scaling and after color palette or color space conversion logic but before color keying multiplexer and before cursor multiplexer.

- After DV, new R = R + (2R - G - B) * FR, where FR is fraction from 0 to 7/8
- After DV, new G = G + (2G - R - B) * FG, where FG is fraction from 0 to 7/8
- After DV, new B = B + (2B - R - G) * FB, where FB is fraction from 0 to 7/8

Offset: 70eh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
18:16	X	B_DV_CONTROL_B: Digital Vibrance control for B

Bit	Reset	Description
10:8	X	B_DV_CONTROL_G: Digital Vibrance control for G
2:0	X	B_DV_CONTROL_R: Digital Vibrance control for R

29.12.58 DC_WIN_B_BLEND_NOKEY_0

Blend Control for this window areas where color key is enabled but the pixel color is not within the color key range (color key not match). This is valid for all overlapping condition but only if there is no overlap with other window with higher priority color key enabled and color key not match.

Class: Display Color Keying and Blending

Color keying and blending of the display windows are done prior to cursor blending.

Cursor always goes on top of the blended windows. Blending is controlled independently on each possible overlap area of the display windows. If 3 windows are enabled there are 7 possible overlap area combinations. For every window in each overlap area combination, color key can be disabled or enabled. Also, for every window in each overlap area combination, there is a corresponding window blend control parameter and a window blend weight parameter. The window blend control parameter is always effective but the window blend weight is not always used. The window blend weight can also be derived from pixel alpha value or from the reverse of other overlapping windows weight.

Color keying has the highest priority for display window blending. Color key consists of a range of color which is searched independently for each windows. If more than 1 windows color keys are enabled then Window A color key has the highest priority, followed by Window B color key, and then followed by Window C color key. Two sets of color key range (Color Key 0 and Color Key 1) can be defined and they are shared for all windows. It is possible to use both color key sets for the same window or for two separate windows.

The two sets of color key ranges should not overlap and if they do the overlap colors are treated as if they are part of Color Key 0 and not part of Color Key 1.

Assuming that there is no overlap with other higher priority window with color key not matched, if a window color key is enabled for any overlap condition and the window pixel is not within the color key range (key not match), then the window pixel will not be blended with other overlapping window pixels but it will be weighted. The weight is not dependent on the overlap condition it is controlled by the same set of parameters for all overlapping condition.

Assuming that there is no overlap with other higher priority window with color key not matched, if a window color key is enabled for a particular overlap condition and the window pixel is within the color key range (key match), then the window pixel will be weighted and blended with other overlapping pixels and this is controlled separately for each overlap condition.

Assuming that there is no overlap with other higher priority window with color key not matched, if a window color key is disabled then the window pixel will be blended with other overlapping pixels and this is controlled separately for each overlap condition.

Display Color Key Parameters

For B4G4R4A4, B5G6R5A, B5G6R5 mode, color key should be compared prior to color conversion to 24-bpp and unused least significant bits of the pixel are filled with zeros.

For palletized mode, color key is compared prior to color palette and the palletized color is compared against the green color key values/mask.

For YUV mode, U and V are offset by +128 before performing the color key comparison.

In all cases, color key is compared prior to horizontal/vertical scaling filter and prior to digital vibrance control. Both upper and lower values are inclusive.

Offset: 70fh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	B_BLEND_WEIGHT1_NOKEY: Window blend weight 1 for color key not match areas. This is used only for alpha weight with 1-bit alpha and alpha value of 1.
15:8	X	B_BLEND_WEIGHT0_NOKEY: Window blend weight 0 for color key not match areas. For alpha weight, this is used for 1-bit alpha with value of 0.
0	X	B_BLEND_CONTROL_NOKEY: Window blend control for color key not match areas. 0 = Fix weight using window blend weight 0 for color key not matched. 1 = Alpha weight using the alpha value. This is only valid if the window color format includes an alpha value. For 1-bit alpha value, if the alpha is 0 window blend weight 0 is used and if alpha is 1, window blend weight 1 is used. 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT

29.12.59 DC_WIN_B_BLEND_1WIN_0

Blend Control for this window area where it does not overlap with other windows.

Offset: 710h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	B_BLEND_WEIGHT1_1WIN: Window blend weight 1 for color key disabled or color key enabled with key matched areas. This is used only for alpha weight with 1-bit alpha and alpha value of 1.
15:8	X	B_BLEND_WEIGHT0_1WIN: Window blend weight 0 for color key disabled or color key enabled with key matched areas. For alpha weight, this is used for 1-bit alpha with value of 0.
2	X	B_BLEND_CONTROL_1WIN: Window blend control in area where it does not overlap with other windows and either color key disabled or color key enabled with key matched. 0 = Fix weight using the window blend weight 0 if color key disabled or color key 0 enabled with key matched, or using the window blend weight 1 if color key 1 enabled with key matched. 1 = Alpha weight using the alpha value. This is only valid if the window color format includes an alpha value. For 1-bit alpha value, if the alpha is 0 window blend weight 0 is used and if alpha is 1, window blend weight 1 is used. 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT
1:0	X	B_CKEY_ENABLE_1WIN: Window color key enable 0 = Color Key 0 and 1 Disable 1 = Color Key 0 Enable 2 = Color Key 1 Enable 3 = Color Key 0 and 1 Enable 0 = NOKEY 1 = CKEY0 2 = CKEY1 3 = CKEY01

29.12.60 DC_WIN_B_BLEND_2WIN_A_0

Blend Control for this window area that overlaps with window A only.

Offset: 711h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	B_BLEND_WEIGHT1_2WIN_A: Window blend weight 1 for color key disabled or color key enabled with key matched areas. This is used only for alpha weight with 1-bit alpha and alpha value of 1.
15:8	X	B_BLEND_WEIGHT0_2WIN_A: Window blend weight 0 for color key disabled or color key enabled with key matched areas. For alpha weight, this is used for 1-bit alpha with value of 0.

Bit	Reset	Description
3:2	X	B_BLEND_CONTROL_2WIN_A: Window blend control in area where either color key disabled or color key enabled with key matched. 0 = Fix weight using the window blend weight 0 if color key disabled or color key 0 enabled with key matched, or using the window blend weight 1 if color key 1 enabled with key matched. 1 = Alpha weight using the alpha value. This is only valid if the window color format includes an alpha value. For 1-bit alpha value, if the alpha is 0 window blend weight 0 is used and if alpha is 1, window blend weight 1 is used. 2 = Dependent weight, in this case, the window blend weight is 1 - the weights of the other window that overlaps with this window. Only 1 of the overlapping windows may have this setting. 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT 2 = DEPENDENT_WEIGHT
1:0	X	B_CKEY_ENABLE_2WIN_A: Window color key enable 0 = Color Key 0 and 1 Disable 1 = Color Key 0 Enable 2 = Color Key 1 Enable 3 = Color Key 0 and 1 Enable 0 = NOKEY 1 = CKEY0 2 = CKEY1 3 = CKEY01

29.12.61 DC_WIN_B_BLEND_2WIN_C_0

Blend Control for this window area that overlaps with window C only.

Offset: 712h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	B_BLEND_WEIGHT1_2WIN_C: Window blend weight 1 for color key disabled or color key enabled with key matched areas. This is used only for alpha weight with 1-bit alpha and alpha value of 1.
15:8	X	B_BLEND_WEIGHT0_2WIN_C: Window blend weight 0 for color key disabled or color key enabled with key matched areas. For alpha weight, this is used for 1-bit alpha with value of 0.
3:2	X	B_BLEND_CONTROL_2WIN_C: Window blend control in area where either color key disabled or color key enabled with key matched. 0 = Fix weight using the window blend weight 0 if color key disabled or color key 0 enabled with key matched, or using the window blend weight 1 if color key 1 enabled with key matched. 1 = Alpha weight using the alpha value. This is only valid if the window color format includes an alpha value. For 1-bit alpha value, if the alpha is 0 window blend weight 0 is used and if alpha is 1, window blend weight 1 is used. 2 = Dependent weight, in this case, the window blend weight is 1 - the weights of the other window that overlaps with this window. Only 1 of the overlapping windows may have this setting. 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT 2 = DEPENDENT_WEIGHT
1:0	X	B_CKEY_ENABLE_2WIN_C: Window color key enable 0 = Color Key 0 and 1 Disable 1 = Color Key 0 Enable 2 = Color Key 1 Enable 3 = Color Key 0 and 1 Enable 0 = NOKEY 1 = CKEY0 2 = CKEY1 3 = CKEY01

29.12.62 DC_WIN_B_BLEND_3WIN_AC_0

Blend Control for this window area that overlaps with windows A & C only.

Offset: 713h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	B_BLEND_WEIGHT1_3WIN_AC: Window blend weight 1 for color key disabled or color key enabled with key matched areas. This is used only for alpha weight with 1-bit alpha and alpha value of 1.
15:8	X	B_BLEND_WEIGHT0_3WIN_AC: Window blend weight 0 for color key disabled or color key enabled with key matched areas. For alpha weight, this is used for 1-bit alpha with value of 0.
3:2	X	B_BLEND_CONTROL_3WIN_AC: Window blend control in area where either color key disabled or color key enabled with key matched. 0 = Fix weight using the window blend weight 0 if color key disabled or color key 0 enabled with key matched, or using the window blend weight 1 if color key 1 enabled with key matched. 1 = Alpha weight using the alpha value. This is only valid if the window color format includes an alpha value. For 1-bit alpha value, if the alpha is 0 window blend weight 0 is used and if alpha is 1, window blend weight 1 is used. 2 = Dependent weight, in this case, the window blend weight is 1 - the weights of the other window that overlaps with this window. Only 1 of the overlapping windows may have this setting. 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT 2 = DEPENDENT_WEIGHT
1:0	X	B_CKEY_ENABLE_3WIN_AC: Window color key enable 0 = Color Key 0 and 1 Disable 1 = Color Key 0 Enable 2 = Color Key 1 Enable 3 = Color Key 0 and 1 Enable 0 = NOKEY 1 = CKEY0 2 = CKEY1 3 = CKEY01

29.12.63 DC_WIN_B_HP_FETCH_CONTROL_0

Window B High Priority Avoidance Fetch Parameters

This register gives extra information to the Memory Controller Client Interface (MCCIF) about the number of memory words that will be fetched per scan line and about the rate at wich those words are consumed. This allows the MCCIF to more accurately arbitrate memory accesses to prevent the use of the High Priority signal. Use of this signal causes all other client accesses to be blocked in preference to the client asserting HP. Obviously, this is a state which is sometime necessary for Display since it is an isochronous client and MUST be serviced in a timely manner. However, use of this signal should be avoided if possible. These parameters help the MCCIF avoid the over-use of HP.

Offset: 714h | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	0x0	B_FETCH_INFO_ENABLE: Enables the sending of the Window B fetch information. For compatibility with earlier devices, this defaults to DISABLE. 0 = DISABLE : This bit should be enabled only for 12-bpp 1 = ENABLE
30:16	X	B_WORDS_PER_LINE: Window B memory fetch words per scan line. This value is in memory fetch words: Multiples of 16 bytes for Tegra 2 Processor Series devices. It is computed as follows: B_WORDS_PER_LINE = (B_SIZE.B_H_SIZE * (bytes per pixel) + 15) >> 4 bytes per pixel is determined by the pixel format.
15:0	X	B_CYCLES_PER_WORD: Window B clock cycles per memory fetch word. The value of this field is essentially a measure of the data consumption rate for window B. It is computed as follows: B_CYCLES_PER_WORD = B_DDA_INCREMENT.B_H_DDA_INCREMENT / (bytes per pixel) Note that the format for this value is a fixed-point fractional value with 8 bits of

Bit	Reset	Description
		integer precision and 8 bits of fractional precision. In other words, it is an '8.8' number. For example, if there is no scaling of the input image, the DDA increment will be 4096. With 32-bit RGBA pixels there will be 4 bytes per pixel, so CYCLES_PER_WORD will be ... 4096 / 4 = 1024, or 4.0 expressed in the 8.8 format. Any scaling performed on the pixels will change the rate at which pixels are consumed. Scaling up will increase the value of DDA increment and will therefore increase the number of cycles between memory fetches. Conversely, scaling down will decrease the value of DDA increment and memory fetches will occur more frequently.

29.13 WINBUF_B Registers

The registers under DC_WINBUF are triple-buffered.

29.13.1 DC_WINBUF_B_START_ADDR_0

Window B Start Address

Overview

START_ADDR, BUF_STRIDE, LINE_STRIDE, ADDR_H_OFFSET, ADDR_V_OFFSET combined, specify the starting address of a window buffer in the memory surface. Generally START_ADDR is programmed with the starting address of the memory surface. H/V_OFFSET specify the address offsets of the pixel at the beginning of the window, with respect to the starting address of the memory surface. (There is an exception to that when memory surface is not well aligned, see 3-ii below.)

Note that "beginning of the window" here means the top-left corner of a window in normal mode, top-right corner in horizontal flipping, bottom-left corner in vertical flipping, and bottom-right in horizontal+vertical flipping.

Starting Address Calculation

These formulae are same for both tiled or linear mode. However, for linear mode, the addresses calculated are real physical addresses, but for tile mode, these addresses will be translated to the real physical addresses by the memory controller client before used.

- When a window is host triggered, starting address of a window is calculated as follows by HW.
  - non-yuv-planar modes:

    starting-address = START_ADDR + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET
  - yuv-planar modes:

    y-starting-address = START_ADDR + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET

    u-starting-address = START_ADDR_U + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2

    v-starting-address = START_ADDR_V + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2 where denom1/denom2 equal to 1 or 2 depending on the actual planar format, derived natively by HW.
- When a window is non-host triggered, starting address of a window buffer is calculated as below.
  - non-yuv-planar mode:

    starting-address = START_ADDR + BUF_STRIDE * buf_index + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET
  - yuv-planar mode:

y-starting-address = START_ADDR + BUF_STRIDE * buf_index + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET

u-starting-address = START_ADDR_U + UV_BUF_STRIDE * buf_index + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2

v-starting-address = START_ADDR_V + UV_BUF_STRIDE * buf_index + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2 where denom1/denom2 equal to 1 or 2 depending on the actual planar format, derived natively by HW.

buf_index is the index transmitted by the triggering module pointing to the first buffer of a frame.

Programming Restrictions

- For tiled address mode:

  Image surface can only aligned to multiples of 256, thus the following restrictions.

  - START_ADDR, START_ADDR_U, START_ADDR_V need to be multiples of 256.
  - BUF_STRIDE, UV_BUF_STRIDE need to be multiples of 256
  - LINE_STRIDE, UV_LINE_STRIDE need to be multiples of 16
  - ADDR_H_OFFSET needs to be multiple of 2 in yuv planar format (the last bit is ignored), but with no restrictions on other color formats.
  - ADDR_V_OFFSET has no restrictions
- For linear address mode:

  Image surface can be aligned 2, 4 or 8 bytes, depending on the color formats.

  As an additional restriction for display, START_ADDR, START_ADDR_U and START_ADDR_V need to be multiples of 16.

  When a surface is not aligned to 16 bytes, program START_ADDR with the memory surface address with its least 4 significant bits zeroed out and add these 4 address bits to the original H_OFFSET. (So the formula in 2-i,ii still hold)

  - For all formats:

    START_ADDR, START_ADDR_U and START_ADDR_V need to be multiples of 16.
  - For 16-bpp formats,

    (START_ADDR+H_OFFSET) need to be multiple of 2. The least significant bit of H_OFFSTE is ignored.
  - For 32-bpp formats,

    (START_ADDR+H_OFFSTE) needs to be multiple of 4. The least two significant bits of H_OFFSTE are ignored.
  - For yuv planar formats:

    BUF_STRIDE, UV_BUF_STRIDE:

    BUF_STRIDE[2:1]=UV_BUF_STRIDE[1:0]

    or as a stricter constraint: BUF_STRIDE be multiple of 8, UV_BUF_STRIDE be multiple of 4.

    LINE_STRIDE, UV_LINE_STRIDE:

    LINE_STRIDE and UV_LINE_STRIDE need to be at least 16.

    LINE_STRIDE needs to be multiple of 8, UV_LINE_STRIDE needs to be multiple of 4.

    ADDR_H_OFFSET:

    Needs to be multiple of 2. If needs to point to odd pixel position, program ADDR_H_OFFSET to be the previous position (or the next position if H-flipped) and program H_INITIAL_DDA bit 12 to 1.

    ADDR_V_OFFSET:

    Needs to be multiple of 2. If needs to point to odd line number, program ADDR_V_OFFSTE to be the previous line number (or next line number if V-flipped) and program V_INITIAL_DDA bit 12 to 1.

- Memory allocation for non-host triggered case

  When a window buffer is not controlled by host (software) then a frame may be stored in multiple buffers. In this case, the buffers must be contiguous in the memory because display will use the same luma or chroma line strides for all lines in the frame.

  Also buffer wraparound must not occur in the middle of the displayed part of the frame. The controlling module will send frame start and frame end indicators (flags) to display module to indicate the beginning and end of frame. Buffer start address is latched when frame start flag is active but the actual buffer start address is not switched until frame end flag is active. In the case where one buffer correspond to one frame then frame start and frame end flag are active everytime a buffer index is sent.

Offset: 800h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	B_START_ADDR: Window B Start Address This is a byte address. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSB are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies start address for the Y plane.

29.13.2 DC_WINBUF_B_START_ADDR_NS_0

Window B Shadowed Start Address

Offset: 801h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	B_START_ADDR_NS: Window B Shadowed Start Address This is ARM set shadow of Start Address.

29.13.3 DC_WINBUF_B_START_ADDR_U_0

Window B Start Address for U plane

Offset: 802h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	B_START_ADDR_U: Window B Start Address for U plane This is a byte address.

29.13.4 DC_WINBUF_B_START_ADDR_U_NS_0

Window B Shadowed Start Address for U plane

Offset: 803h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	B_START_ADDR_U_NS: Window B Shadowed Start Address for U plane This is ARM set shadow register of U start address

29.13.5 DC_WINBUF_B_START_ADDR_V_0

Window B Start Address for V plane

Offset: 804h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	B_START_ADDR_V: Window B Start Address for V plane This is a byte address.

29.13.6 DC_WINBUF_B_START_ADDR_V_NS_0

Window B Shadowed Start Address for V plane

Offset: 805h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	B_START_ADDR_V_NS: Window B Shadowed Start Address for V plane This is ARM set shadow register of U start address

29.13.7 DC_WINBUF_B_ADDR_H_OFFSET_0

Window B Horizontal address offset

Offset: 806h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	B_ADDR_H_OFFSET: Window B Horizontal address offset This is a byte address. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSB are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies horizontal offset of Y plane. The horizontal offsets of U/V plane is derived by HW.

29.13.8 DC_WINBUF_B_ADDR_H_OFFSET_NS_0

Window B Shadowed Horizontal address offset

Offset: 807h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	B_ADDR_H_OFFSET_NS: Window B Shadowed Horizontal address offset This is ARM set shadow of ADDR_H_OFFSET

29.13.9 DC_WINBUF_B_ADDR_V_OFFSET_0

Window B Vertical address offset

Offset: 808h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	B_ADDR_V_OFFSET: Window B Vertical address offset This is a byte address. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSB are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies vertical offset of Y plane. Vertical offsets of U/V

Bit	Reset	Description
		plane is derived by HW.

29.13.10 DC_WINBUF_B_ADDR_V_OFFSET_NS_0

Window B Shadowed Vertical address offset

Offset: 809h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	B_ADDR_V_OFFSET_NS: Window B Shadowed Vertical address offset This is ARM set shadow of ADDR_V_OFFSET

29.13.11 DC_WINBUF_B_UFLOW_STATUS_0

Window B FIFO Underflow Status Register

Offset: 80ah | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
30	X	COUNT_OFLOW: Flag bit that indicates that the underflow event counter has overflowed. There were too many events. If COUNT_OFLOW is set, UFLOW_COUNT is meaningless. Cleared on write.
23:0	X	UFLOW_COUNT: Underflow count. This field indicates the number of contiguous groups of output pixels for which there was no data in the FIFO. For example, if the valid from the FIFO is low for 10 consecutive cycles and then goes high, the counter will increment by one. Reset to zero on write.

29.14 Window C (WIN_C) Registers

These registers control window C parameters

29.14.1 DC_WINC_C_COLOR_PALETTE_0

Window C Color Palette

This is used for palletized data format (color depth of 8-bpp or less) or for gamma correction for non-palletized data formats (color depth of more than 8-bpp). Each window has its own color palette which consists of three 256x8 register file which can be written by host and indexed (read) by the window.

For palletized data format less than 8-bpp the pixel data is aligned to least significant bits of the palette index (address) and the remaining upper bits are filled with the corresponding bits of the Palette Color Extension. For example, for 4-bpp mode, the pixel data occupies bits 3-0 of the palette index and bits 7-4 of the palette index are set to bits 7-4 of the Palette Color Extension. Host read is assumed to be not needed - software can cache the color palette in system memory. This is an array of 256 identical register entries; the register fields below apply to each entry.

Color palette

Offset: 500h..5ffh | Read/Write: WO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	C_COLOR_PALETTE_B: Blue Color Palette

Bit	Reset	Description
15:8	X	C_COLOR_PALETTE_G: Green Color Palette
7:0	X	C_COLOR_PALETTE_R: Red Color Palette

29.14.2 DC_WINC_C_PALETTE_COLOR_EXT_0

Window C Palette Color Extension

Palette extension for 1-bpp, 2-bpp, and 4-bpp. These bits provide the upper most significant bits for indexing the color palette. Supported for window A only.

XXX should ifdef for window A, but currently window B spec is assumed to be a superset.

Offset: 600h | Read/Write: R/W | Reset: 0bxxxxxxx

Bit	Reset	Description
7:1	X	C_PALETTE_COLOR_EXT: Window C Palette Color Extension bits 7-1 are used for 1-bpp mode bits 7-2 are used for 2-bpp mode bits 7-4 are used for 4-bpp mode

29.14.3 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 value ranging from -4 to 3.
- Coefficients 1 and 4 are 5-bit signed value ranging from -16 to 15.
- Coefficients 2 and 3 are 8-bit unsigned value ranging from 0 to 128.
- Coefficient 0 is the multiplier for the earliest pixel (P0) in the group of 6-pixel and coefficient 5 is the multiplier for the latest pixel (P5) in the group. The output pixel positional phase is defined as centered in P2 if the positional phase is 0 or proportionally in between P2 and P3 if the positional phase is larger than 0.

Sum of all coefficients for each phase should be 128 typically and software should never program the the sum of all coefficients for a phase to be more than 128.

For each horizontal positional phase, the 6 filter coefficients require 32 reg bits. Note that color value ranges from 0 to 255 for Y, R, G, B and -128 to 127 for U and V.

Offset: 601h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	C_H_FILTER_P00C5: Phase 00 coefficient 5 (typically 0)
28:24	X	C_H_FILTER_P00C4: Phase 00 coefficient 4 (typically 0)
23:16	X	C_H_FILTER_P00C3: Phase 00 coefficient 3 (typically 0)
15:8	X	C_H_FILTER_P00C2: Phase 00 coefficient 2 (typically 128)
7:3	X	C_H_FILTER_P00C1: Phase 00 coefficient 1 (typically 0)
2:0	X	C_H_FILTER_P00C0: Phase 00 coefficient 0 (typically 0)

29.14.4 DC_WINC_C_H_FILTER_P01_0

Window C Horizontal Filter phase 01

Offset: 602h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	C_H_FILTER_P01C5: Phase 01 coefficient 5 (typically 1)
28:24	X	C_H_FILTER_P01C4: Phase 01 coefficient 4 (typically -2)
23:16	X	C_H_FILTER_P01C3: Phase 01 coefficient 3 (typically 8)
15:8	X	C_H_FILTER_P01C2: Phase 01 coefficient 2 (typically 124)
7:3	X	C_H_FILTER_P01C1: Phase 01 coefficient 1 (typically -4)
2:0	X	C_H_FILTER_P01C0: Phase 01 coefficient 0 (typically 1)

29.14.5 DC_WINC_C_H_FILTER_P02_0

Window C Horizontal Filter phase 02

Offset: 603h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	C_H_FILTER_P02C5: Phase 02 coefficient 5 (typically 1)
28:24	X	C_H_FILTER_P02C4: Phase 02 coefficient 4 (typically -5)
23:16	X	C_H_FILTER_P02C3: Phase 02 coefficient 3 (typically 17)
15:8	X	C_H_FILTER_P02C2: Phase 02 coefficient 2 (typically 122)
7:3	X	C_H_FILTER_P02C1: Phase 02 coefficient 1 (typically -8)
2:0	X	C_H_FILTER_P02C0: Phase 02 coefficient 0 (typically 1)

29.14.6 DC_WINC_C_H_FILTER_P03_0

Window C Horizontal Filter phase 03

Offset: 604h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	C_H_FILTER_P03C5: Phase 03 coefficient 5 (typically 2)
28:24	X	C_H_FILTER_P03C4: Phase 03 coefficient 4 (typically -7)
23:16	X	C_H_FILTER_P03C3: Phase 03 coefficient 3 (typically 27)
15:8	X	C_H_FILTER_P03C2: Phase 03 coefficient 2 (typically 115)
7:3	X	C_H_FILTER_P03C1: Phase 03 coefficient 1 (typically -11)
2:0	X	C_H_FILTER_P03C0: Phase 03 coefficient 0 (typically 2)

29.14.7 DC_WINC_C_H_FILTER_P04_0

Window C Horizontal Filter phase 04

Offset: 605h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	C_H_FILTER_P04C5: Phase 04 coefficient 5 (typically 2)
28:24	X	C_H_FILTER_P04C4: Phase 04 coefficient 4 (typically -9)
23:16	X	C_H_FILTER_P04C3: Phase 04 coefficient 3 (typically 37)
15:8	X	C_H_FILTER_P04C2: Phase 04 coefficient 2 (typically 109)
7:3	X	C_H_FILTER_P04C1: Phase 04 coefficient 1 (typically -13)
2:0	X	C_H_FILTER_P04C0: Phase 04 coefficient 0 (typically 2)

29.14.8 DC_WINC_C_H_FILTER_P05_0

Window C Horizontal Filter phase 05

Offset: 606h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	C_H_FILTER_P05C5: Phase 05 coefficient 5 (typically 2)
28:24	X	C_H_FILTER_P05C4: Phase 05 coefficient 4 (typically -11)
23:16	X	C_H_FILTER_P05C3: Phase 05 coefficient 3 (typically 47)
15:8	X	C_H_FILTER_P05C2: Phase 05 coefficient 2 (typically 102)
7:3	X	C_H_FILTER_P05C1: Phase 05 coefficient 1 (typically -15)
2:0	X	C_H_FILTER_P05C0: Phase 05 coefficient 0 (typically 3)

29.14.9 DC_WINC_C_H_FILTER_P06_0

Window C Horizontal Filter phase 06

Offset: 607h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	C_H_FILTER_P06C5: Phase 06 coefficient 5 (typically 3)
28:24	X	C_H_FILTER_P06C4: Phase 06 coefficient 4 (typically -13)
23:16	X	C_H_FILTER_P06C3: Phase 06 coefficient 3 (typically 56)
15:8	X	C_H_FILTER_P06C2: Phase 06 coefficient 2 (typically 94)
7:3	X	C_H_FILTER_P06C1: Phase 06 coefficient 1 (typically -15)
2:0	X	C_H_FILTER_P06C0: Phase 06 coefficient 0 (typically 3)

29.14.10 DC_WINC_C_H_FILTER_P07_0

Window C Horizontal Filter phase 07

Offset: 608h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	C_H_FILTER_P07C5: Phase 07 coefficient 5 (typically 3)
28:24	X	C_H_FILTER_P07C4: Phase 07 coefficient 4 (typically -14)
23:16	X	C_H_FILTER_P07C3: Phase 07 coefficient 3 (typically 67)
15:8	X	C_H_FILTER_P07C2: Phase 07 coefficient 2 (typically 85)
7:3	X	C_H_FILTER_P07C1: Phase 07 coefficient 1 (typically -16)
2:0	X	C_H_FILTER_P07C0: Phase 07 coefficient 0 (typically 3)

29.14.11 DC_WINC_C_H_FILTER_P08_0

Window C Horizontal Filter phase 08

Offset: 609h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	C_H_FILTER_P08C5: Phase 08 coefficient 5 (typically 3)
28:24	X	C_H_FILTER_P08C4: Phase 08 coefficient 4 (typically -15)
23:16	X	C_H_FILTER_P08C3: Phase 08 coefficient 3 (typically 76)
15:8	X	C_H_FILTER_P08C2: Phase 08 coefficient 2 (typically 76)
7:3	X	C_H_FILTER_P08C1: Phase 08 coefficient 1 (typically -15)
2:0	X	C_H_FILTER_P08C0: Phase 08 coefficient 0 (typically 3)

29.14.12 DC_WINC_C_H_FILTER_P09_0

Window C Horizontal Filter phase 09

Offset: 60ah | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	C_H_FILTER_P09C5: Phase 09 coefficient 5 (typically 3)
28:24	X	C_H_FILTER_P09C4: Phase 09 coefficient 4 (typically -16)
23:16	X	C_H_FILTER_P09C3: Phase 09 coefficient 3 (typically 85)
15:8	X	C_H_FILTER_P09C2: Phase 09 coefficient 2 (typically 67)
7:3	X	C_H_FILTER_P09C1: Phase 09 coefficient 1 (typically -14)
2:0	X	C_H_FILTER_P09C0: Phase 09 coefficient 0 (typically 3)

29.14.13 DC_WINC_C_H_FILTER_P0A_0

Window C Horizontal Filter phase 0A

Offset: 60bh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	C_H_FILTER_P0AC5: Phase 0A coefficient 5 (typically 3)
28:24	X	C_H_FILTER_P0AC4: Phase 0A coefficient 4 (typically -15)
23:16	X	C_H_FILTER_P0AC3: Phase 0A coefficient 3 (typically 94)
15:8	X	C_H_FILTER_P0AC2: Phase 0A coefficient 2 (typically 56)
7:3	X	C_H_FILTER_P0AC1: Phase 0A coefficient 1 (typically -13)
2:0	X	C_H_FILTER_P0AC0: Phase 0A coefficient 0 (typically 3)

29.14.14 DC_WINC_C_H_FILTER_P0B_0

Window C Horizontal Filter phase 0B

Offset: 60ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	C_H_FILTER_P0BC5: Phase 0B coefficient 5 (typically 3)
28:24	X	C_H_FILTER_P0BC4: Phase 0B coefficient 4 (typically -15)
23:16	X	C_H_FILTER_P0BC3: Phase 0B coefficient 3 (typically 102)
15:8	X	C_H_FILTER_P0BC2: Phase 0B coefficient 2 (typically 47)
7:3	X	C_H_FILTER_P0BC1: Phase 0B coefficient 1 (typically -11)
2:0	X	C_H_FILTER_P0BC0: Phase 0B coefficient 0 (typically 2)

29.14.15 DC_WINC_C_H_FILTER_P0C_0

Window C Horizontal Filter phase 0C

Offset: 60dh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	C_H_FILTER_P0CC5: Phase 0C coefficient 5 (typically 2)
28:24	X	C_H_FILTER_P0CC4: Phase 0C coefficient 4 (typically -13)
23:16	X	C_H_FILTER_P0CC3: Phase 0C coefficient 3 (typically 109)
15:8	X	C_H_FILTER_P0CC2: Phase 0C coefficient 2 (typically 37)
7:3	X	C_H_FILTER_P0CC1: Phase 0C coefficient 1 (typically -9)
2:0	X	C_H_FILTER_P0CC0: Phase 0C coefficient 0 (typically 2)

29.14.16 DC_WINC_C_H_FILTER_P0D_0

Window C Horizontal Filter phase 0D

Offset: 60eh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	C_H_FILTER_P0DC5: Phase 0D coefficient 5 (typically 2)
28:24	X	C_H_FILTER_P0DC4: Phase 0D coefficient 4 (typically -11)
23:16	X	C_H_FILTER_P0DC3: Phase 0D coefficient 3 (typically 115)
15:8	X	C_H_FILTER_P0DC2: Phase 0D coefficient 2 (typically 27)
7:3	X	C_H_FILTER_P0DC1: Phase 0D coefficient 1 (typically -7)
2:0	X	C_H_FILTER_P0DC0: Phase 0D coefficient 0 (typically 2)

29.14.17 DC_WINC_C_H_FILTER_P0E_0

Window C Horizontal Filter phase 0E

Offset: 60fh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	C_H_FILTER_P0EC5: Phase 0E coefficient 5 (typically 1)
28:24	X	C_H_FILTER_P0EC4: Phase 0E coefficient 4 (typically -8)
23:16	X	C_H_FILTER_P0EC3: Phase 0E coefficient 3 (typically 122)
15:8	X	C_H_FILTER_P0EC2: Phase 0E coefficient 2 (typically 17)
7:3	X	C_H_FILTER_P0EC1: Phase 0E coefficient 1 (typically -5)
2:0	X	C_H_FILTER_P0EC0: Phase 0E coefficient 0 (typically 1)

29.14.18 DC_WINC_C_H_FILTER_P0F_0

Window C Horizontal Filter phase 0F

Offset: 610h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	C_H_FILTER_P0FC5: Phase 0F coefficient 5 (typically 1)
28:24	X	C_H_FILTER_P0FC4: Phase 0F coefficient 4 (typically -4)
23:16	X	C_H_FILTER_P0FC3: Phase 0F coefficient 3 (typically 124)
15:8	X	C_H_FILTER_P0FC2: Phase 0F coefficient 2 (typically 8)
7:3	X	C_H_FILTER_P0FC1: Phase 0F coefficient 1 (typically -2)
2:0	X	C_H_FILTER_P0FC0: Phase 0F coefficient 0 (typically 1)

29.14.19 DC_WINC_C_CSC_YOF_0

Window C CSC Y Offset

Color Space Conversion coefficients

The CSC can be used for YUV to RGB conversion with brightness and hue/saturation control. The CSC can only be enabled for window C controlled by CSC_ENABLE register bits.

For Y color, the Y offset is applied first and saturation (clipping) is performed immediately after the Y offset is applied.

- R = sat(KYRGB * sat(Y + YOF) + KUR * U + KVR * V)
- G = sat(KYRGB * sat(Y + YOF) + KUG * U + KVG * V)
- B = sat(KYRGB * sat(Y + YOF) + KUB * U + KVB * V)

Saturation and rounding is performed in the range of 0 to 255 for the above equations.

Typical values are:

- YOF = -16.000, KYRGB = 1.1644
- KUR = 0.0000, KVR = 1.5960
- KUG = -0.3918, KVG = -0.8130
- KUB = 2.0172, KVB = 0.0000

KUR and KVB are typically 0.0000 but they may be programmed non-zero for hue rotation.

The CSC can also take RGB input, in which case YOF, KVB, KUG, KUR should be programmed to 0 and KYRGB will be forced to 0 by the hardware for generating R and B. KYRGB will not be forced to 0 for generating G. KVR, KYRGB, and KUB can be programmed to 1.0 or used as gain control for R, G, B correspondingly. Note that color value ranges from 0 to 255 for Y, R, G, B and -128 to 127 for U and V.

Offset: 611h | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	C_CSC_YOF: Y Offset in s.7.0 format

29.14.20 DC_WINC_C_CSC_KYRGB_0

Window C CSC Y Coefficient (gain) for RGB

Offset: 612h | Read/Write: R/W | Reset: 0bxxxxxxxxxx

Bit	Reset	Description
9:0	X	C_CSC_KYRGB: Y Gain for R, G, B colors in 2.8 format

29.14.21 DC_WINC_C_CSC_KUR_0

Window C CSC U coefficient for R

Offset: 613h | Read/Write: R/W | Reset: 0bxxxxxxxxxxx

Bit	Reset	Description
10:0	X	C_CSC_KUR: U coefficients for R in s.2.8 format

29.14.22 DC_WINC_C_CSC_KVR_0

Window C CSC V coefficient for R

Offset: 614h | Read/Write: R/W | Reset: 0bxxxxxxxxxxx

Bit	Reset	Description
10:0	X	C_CSC_KVR: V coefficients for R in s.2.8 format

29.14.23 DC_WINC_C_CSC_KUG_0

Window C CSC U coefficient for G

Offset: 615h | Read/Write: R/W | Reset: 0bxxxxxxxxxx

Bit	Reset	Description
9:0	X	C_CSC_KUG: U coefficients for G in s.1.8 format

29.14.24 DC_WINC_C_CSC_KVG_0

Window C CSC V coefficient for G

Offset: 616h | Read/Write: R/W | Reset: 0bxxxxxxxxxx

Bit	Reset	Description
9:0	X	C_CSC_KVG: V coefficients for G in s.1.8 format

29.14.25 DC_WINC_C_CSC_KUB_0

Window C CSC U coefficient for B

Offset: 617h | Read/Write: R/W | Reset: 0bxxxxxxxxxxx

Bit	Reset	Description
10:0	X	C_CSC_KUB: U coefficients for B in s.2.8 format

29.14.26 DC_WINC_C_CSC_KVB_0

Window C CSC V coefficient for B

Offset: 618h | Read/Write: R/W | Reset: 0bxxxxxxxxxxx

Bit	Reset	Description
10:0	X	C_CSC_KVB: V coefficients for B in s.2.8 format

Vertical scaling filter coefficients

Vertical scaling filter is a 2-tap filter with 4-bit positional phase. Coefficients 0 and 1 are 8-bit unsigned value ranging from 0 to 128.

Coefficient 0 is the multiplier for the earlier pixel (P0) in the group of 2-pixel and coefficient 1 is the multiplier for the later pixel (P1) in the group. The output pixel positional phase is defined as centered in P0 if the positional phase is 0 or proportionally in between P0 and P1 if the positional phase is larger than 0.

Sum of all coefficients for each phase should be 128 typically therefore coefficient 1 can be calculated from (1 - coefficient 0) and therefore only coefficient 0 is programmed.

For each vertical positional phase, the filter coefficient requires 8 reg bits. Note that color value ranges from 0 to 255 for Y, R, G, B and -128 to 127 for U and V.

The registers under DC_WIN are double buffered.

29.14.27 DC_WIN_C_WIN_OPTIONS_0

Window C Options

Class: Display Window Settings

Display Window C parameters

Offset: 700h | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxx0xxxxxxxxxxxxxxx

Bit	Reset	Description
30	0x0	C_WIN_ENABLE: Window C Window enable 0 = DISABLE 1 = ENABLE
22	X	C_YUV_RANGE_EXPAND: Window C Enable range expansion in the cases where RANGEREDFRM is 1 from mpd. Formula: Y = clip((Y-128)*2 + 128); Cb = clip((Cb-128)*2 + 128); Cr = clip((Cr-128)*2 + 128); where clip() function clips between 0 and 255. 0= disable 1= enable 0 = DISABLE 1 = ENABLE
20	X	C_DV_ENABLE: Window C Digital Vibrance Enable 0 = DISABLE 1 = ENABLE
18	X	C_CSC_ENABLE: Window C Color Space Conversion Enable This controls the color space conversion and should be enabled for YCbCr/YUV color modes for conversion to B8G8R8 and for hue and saturation control. This can also be used for gain control for RGB color modes. 0 = DISABLE 1 = ENABLE
16	0x0	C_CP_ENABLE: Window C Color Palette Enable This controls the color palette and should be enabled for palletized color modes. For non-palletized color modes, the color palette can be enabled for gamma correction. 0 = DISABLE 1 = ENABLE
8	X	C_H_FILTER_ENABLE: Window C H Filter Enable This controls H scaling filter and is effective only for non-palletized color modes. 0 = DISABLE 1 = ENABLE
6	X	C_COLOR_EXPAND: Window C 12/15/16/18-to-24 bpp color expansion This bit should be enabled only for 12-bpp B4G4R4A4, 15-bpp B5G5R5A, 16-bpp B5G6R5, 18-bpp B6G6R6 color modes. If enabled the color conversion is performed prior to horizontal scaling. 0 = DISABLE 1 = ENABLE
2	X	C_V_DIRECTION: Window C Vertical (Y) drawing Direction 0 = INCREMENT

Bit	Reset	Description
		1 = DECREMENT
0	X	C_H_DIRECTION: Window C Horizontal (X) drawing Direction 0 = INCREMENT 1 = DECREMENT

29.14.28 DC_WIN_C_BYTE_SWAP_0

Window C Byte Swap

Offset: 701h | Read/Write: R/W | Reset: 0bxx

Bit	Reset	Description
1:0	X	C_BYTE_SWAP: Window C Byte Swap This controls byte swap of frame data read from memory prior to any data processing in the display module. 00= no byte swap (3 2 1 0) 01= byte swap for each 2-byte word (2 3 0 1) 10= byte swap for each 4-byte word (0 1 2 3) 11= word swap for each 4-byte word (1 0 3 2) 0 = NOSWAP 1 = SWAP2 2 = SWAP4 3 = SWAP4HW

29.14.29 DC_WIN_C_BUFFER_CONTROL_0

Window C Buffer Control

Offset: 702h | Read/Write: R/W | Reset: 0b000

Bit	Reset	Description
2:0	0x0	C_BUFFER_CONTROL: Window C Buffer Control 0= Host (software) controlled 1= Video Input controlled 2= Encoder Pre-Processor controlled 3= MPEG Encoder controlled 4= StretchBLT or 2D other= reserved If window buffer selection is not controlled by host (software) then buffer start indexes are sent by the respective module specified by this parameter, and in this case, the buffer start address registers are used to specify frame stride and buffer offset for the calculated start address. 0 = HOST 1 = VI 2 = EPP 4 = SB2D 3 = MPEGE

29.14.30 DC_WIN_C_COLOR_DEPTH_0

Window C Color Depth For YCbCr data format, Cb and Cr are 8-bit unsigned values. For YUV data format, U and V are 8-bit signed values. YCbCr422R is similar to YCbCr422P but the Cb and Cr are shared vertically.

YUV422R is similar to YUV422P but the U and V are shared vertically. YCbCr422RA is same as YCbCr422R in memory and YUV422RA is same as YUV422R in memory but while reading from memory, for YCbCr422RA and YUV422RA, chroma averaging is applied for each pixel pair so that they can be processed as YUV422 by the display pipeline.

For YCbCr422R and YUV422R, every other chroma pixels are not used (discarded) by the display pipeline. B6x2G6x2R6x2A8 is similar to B8G8R6A8 but with the 2 lsb zeroed out. R6x2G6x2B6x2A8 is similar to R8G8B6A8 but with the 2 lsb zeroed out.

Offset: 703h | Read/Write: R/W | Reset: 0bxxxxx

Bit	Reset	Description
4:0	X	C_COLOR_DEPTH: Window C Color Depth Supported color depths are: P8 = 8-bpp (palletized) B4G4R4A4 = 12-bpp B4G4R4 B5G5R5A = 15-bpp B5G5R5 AB5G5R5 = 15-bpp B5G5R5 B5G6R5 = 16-bpp B5G6R5 B8G8R8A8 = 32-bpp B8G8R8A8 R8G8B8A8 = 32-bpp R8G8B8A8 B6x2G6x2R6x2A8 = 32-bpp B6G6R6A8 R6x2G6x2B6x2A8 = 32-bpp R6G6B6A8 YCbCr422 = 16-bpp YCbCr422 packed YUV422 = 16-bpp YUV422 YCbCr420P = 16-bpp YCbCr420 planar YUV420P = 16-bpp YUV420 planar YCbCr422P = 16-bpp YCbCr422 planar YUV422P = 16-bpp YUV422 planar YCbCr422R = 16-bpp YCbCr422 rotated planar YUV422R = 16-bpp YUV422 rotated planar YCbCr422RA= 16-bpp YCbCr422 rotated planar with chroma averaging YUV422RA = 16-bpp YUV422 rotated planar with chroma averaging 0 = P1 1 = P2 2 = P4 3 = P8 4 = B4G4R4A4 5 = B5G5R5A 6 = B5G6R5 7 = AB5G5R5 12 = B8G8R8A8 13 = R8G8B8A8 14 = B6x2G6x2R6x2A8 15 = R6x2G6x2B6x2A8 16 = YCbCr422 17 = YUV422 18 = YCbCr420P 19 = YUV420P 20 = YCbCr422P 21 = YUV422P 22 = YCbCr422R 23 = YUV422R 24 = YCbCr422RA 25 = YUV422RA

29.14.31 DC_WIN_C_POSITION_0

Window C Position

This register defines H position and size of Window C after scaling (if there is any)

Offset: 704h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	C_V_POSITION: Window C V Position This is specified with respect to the top edge of active display area.
12:0	X	C_H_POSITION: Window C H Position This is specified with respect to the left edge of active display area.

29.14.32 DC_WIN_C_SIZE_0

Window C Size

This register defines V position and size of Window C after scaling (if there is any)

Note: Programming on window size should guarantee the whole window is inside the active area, otherwise extra rows/columns will be fetched and discarded which affects performance.

Programming on window size should guarantee the whole window is inside the active area, otherwise extra rows/columns will be fetched and discarded which affects performance.

Offset: 705h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	C_V_SIZE: Window C V Size (lines) This is the vertical size after scaling.
12:0	X	C_H_SIZE: Window C H Size (pixels) This is the horizontal size after scaling.

29.14.33 DC_WIN_C_PRESCALED_SIZE_0

Window C Pre-scaled Size

This register defines Window C pre-scaled size.

The H pre-scaled size is needed to determine how many bytes to fetch from memory per line and this parameter must be programmed exactly as needed taking into account the scaling factor. For planar YUV or YCbCr data formats, this parameter refer to the H pre-scaled of the Y plane.

The total number of lines to be fetched from memory is determined by post-scale V size but V pre-scaled size is needed to 'clamp' the last valid line if the vertical DDA is exactly or slightly beyond the specified V pre-scaled size.

Design Note: H pre-scaled size ideally should be in terms of pixel but then hardware needs to convert this precisely to bytes to determine the amount of data to request from memory.

This could be a risky calculation - maybe this should be made optional on whether we use internal hardware to calculate or left it to software to calculate.

Offset: 706h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	C_V_PRESCALED_SIZE: Window C V Pre-scaled Size (lines) In 420P/422R/422RA formats, it must be even.
14:0	X	C_H_PRESCALED_SIZE: Window C H Pre-scaled Size (bytes) In 420P and 422P formats, it must be even.

29.14.34 DC_WIN_C_H_INITIAL_DDA_0

Window C H Initial DDA

Design Note: the first pixel of pre-scaled image is always used to output the first pixel so essentially this is the same as forcing the H Initial DDA integer portion to 1 initially even though user typically programs this to 0. If it makes the implementation easier, it is possible to force software to program the Initial DDA integer portion to 1. Similarly with the V Initial DDA.

Offset: 707h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	C_H_INITIAL_DDA: Window C H Initial DDA (4.12) This is typically programmed to 0.0

29.14.35 DC_WIN_C_V_INITIAL_DDA_0

Window C V Initial DDA

Offset: 708h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	C_V_INITIAL_DDA: Window C V Initial DDA (4.12) This is typically programmed to 0.0 for both non-interlaced and interlaced sources.

29.14.36 DC_WIN_C_DDA_INCREMENT_0

Window C DDA Increment

DDA increment is typically calculated by dividing (Pre-scaled size in pixels - 1) by (Post-scaled size in pixels - 1). The result should be rounded up and expressed as 4.12 format (4-bit integer and 12-bit fraction). For non-filtered image this value can be slightly larger so that it is not missing the last row/column. Reference programming values (H and V should be calculated separately depending on filter on/off and sizes):

- Filter on: min(round((prescaled_size_in_pixels - 1) * 0x1000 /(post_scaled_size_in_pixels - 1)), MAX)

- Filter off: min(round(prescaled_size_in_pixels * 0x1000 / post_scaled_size_in_pixels - 1) - 0.5), MAX)

Where the value of MAX is as follows:

For V_DDA_INCREMENT: 15.0 (0xF000)

For H_DDA_INCREMETN: 4.0 (0x4000) for 2 Bytes/pix formats.

8.0 (0x8000) for 4 Bytes/pix formats.

They are theoretically the biggest values that guarantees not displaying beyond an image boundary. If the DDA increment is less than 1.0 then image will be up-scaled and if DDA increment is more than 1.0 then image will be down-scaled.

Offset: 709h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	C_V_DDA_INCREMENT: Window C Vertical DDA Increment (4.12) This should be set to 1.0 if there is no scaling. Maximum value is 15.0 regardless of the number of bytes per pixel.
15:0	X	C_H_DDA_INCREMENT: Window C Horizontal DDA Increment (4.12) This should be set to 1.0 if there is no scaling. The maximum value for downscaling depends on the number of bytes per pixel. For 4-byte/pixel modes (32-bpp) the maximum value is 4.0 and for all other modes the maximum value is 8.0.

29.14.37 DC_WIN_C_LINE_STRIDE_0

Window C Line Stride

Offset: 70ah | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	C_UV_LINE_STRIDE: Window C Line Stride for Chroma This is stride (in bytes) for planar YUV or YCbCr data formats for the chroma plane, with the restriction that it must be programmed to be multiples of 4 (16 if tiled or in horizontal flipping) This is not used (ignored) for other non-planar data formats.

Bit	Reset	Description
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.

29.14.38 DC_WIN_C_BUF_STRIDE_0

Window C Buffer stride

Offset: 70bh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	C_BUF_STRIDE: Window C Buffer stride Buffer stride is used to calculate the buffer addresses when the window is triggered by non-host modules. Refer to the comment of START_ADDR for programming guide. For YUV planar pixel format, this specifies buffer stride for the Y plane. The value is in bytes.

29.14.39 DC_WIN_C_UV_BUF_STRIDE_0

Window C Buffer stride for U/V plane

Offset: 70ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	C_UV_BUF_STRIDE: Window C This value is in bytes. For YUV planar pixel format, this specifies buffer stride for the U/V plane.

29.14.40 DC_WIN_C_BUFFER_ADDR_MODE_0

Memory Controller Tiling definitions

Offset: 70dh | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxx0

Bit	Reset	Description
16	0x0	C_UV_TILE_MODE: Window C Memory surface tiling mode For YUV planar pixel format, this specifies tiling mode for the U/V plane. 0 = LINEAR 1 = TILED
0	0x0	C_TILE_MODE: Window C Memory surface tiling mode For YUV planar pixel format, this specifies tiling mode for the Y plane. 0 = LINEAR 1 = TILED

29.14.41 DC_WIN_C_DV_CONTROL_0

Window C Digital Vibrance Control

If enabled, Digital Vibrance is applied after H and V scaling and after color palette or color space conversion logic but before color keying multiplexer and before cursor multiplexer.

- After DV, new R = R + (2R - G - B) * FR, where FR is fraction from 0 to 7/8

- After DV, new G = G + (2G - R - B) * FG, where FG is fraction from 0 to 7/8
- After DV, new B = B + (2B - R - G) * FB, where FB is fraction from 0 to 7/8

Offset: 70eh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
18:16	X	C_DV_CONTROL_B: Digital Vibrance control for B
10:8	X	C_DV_CONTROL_G: Digital Vibrance control for G
2:0	X	C_DV_CONTROL_R: Digital Vibrance control for R

29.14.42 DC_WIN_C_BLEND_NOKEY_0

Blend Control for this window area where color key is enabled but the pixel color is not within the color key range (color key not match). This is valid for all overlapping condition but only if there is no overlap with other window with higher priority color key enabled and color key not match.

Class: Display Color Keying and Blending

Color keying and blending of the display windows are done prior to cursor blending. Cursor always goes on top of the blended windows. Blending is controlled independently on each possible overlap area of the display windows. If 3 windows are enabled there are 7 possible overlap area combinations. For every window in each overlap area combination, color key can be disabled or enabled. Also, for every window in each overlap area combination, there is a corresponding window blend control parameter and a window blend weight parameter. The window blend control parameter is always effective but the window blend weight is not always used. The window blend weight can also be derived from pixel alpha value or from the reverse of other overlapping windows weight.

Color keying has the highest priority for display window blending. Color key consists of a range of color which is searched independently for each window. If more than 1 windows color keys are enabled then Window A color key has the highest priority, followed by Window B color key, and then followed by Window C color key. Two sets of color key range (Color Key 0 and Color Key 1) can be defined and they are shared for all windows. It is possible to use both color key sets for the same window or for two separate windows.

The two sets of color key ranges should not overlap and if they do the overlap colors are treated as if they are part of Color Key 0 and not part of Color Key 1.

Assuming that there is no overlap with other higher priority window with color key not matched, if a window color key is enabled for any overlap condition and the window pixel is not within the color key range (key not match), then the window pixel will not be blended with other overlapping window pixels but it will be weighted. The weight is not dependent on the overlap condition it is controlled by the same set of parameters for all overlapping condition.

Assuming that there is no overlap with other higher priority window with color key not matched, if a window color key is enabled for a particular overlap condition and the window pixel is within the color key range (key match), then the window pixel will be weighted and blended with other overlapping pixels and this is controlled separately for each overlap condition.

Assuming that there is no overlap with other higher priority window with color key not matched, if a window color key is disabled then the window pixel will be blended with other overlapping pixels and this is controlled separately for each overlap condition.

Display Color Key Parameters

For B4G4R4A4, B5G6R5A, B5G6R5 mode, color key should be compared prior to color conversion to 24-bpp and unused least significant bits of the pixel are filled with zeros.

For palletized mode, color key is compared prior to color palette and the palletized color is compared against the green color key values/mask.

For YUV mode, U and V are offset by +128 before performing the color key comparison.

In all cases, color key is compared prior to horizontal/vertical scaling filter and prior to digital vibrance control.

Both upper and lower values are inclusive.

Offset: 70fh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	C_BLEND_WEIGHT1_NOKEY: Window blend weight 1 for color key not match areas. This is used only for alpha weight with 1-bit alpha and alpha value of 1.
15:8	X	C_BLEND_WEIGHT0_NOKEY: Window blend weight 0 for color key not match areas. For alpha weight, this is used for 1-bit alpha with value of 0.
0	X	C_BLEND_CONTROL_NOKEY: Window blend control for color key not match areas. 0 = Fix weight using window blend weight 0 for color key not matched. 1 = Alpha weight using the alpha value. This is only valid if the window color format includes an alpha value. For 1-bit alpha value, if the alpha is 0 window blend weight 0 is used and if alpha is 1, window blend weight 1 is used. 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT

29.14.43 DC_WIN_C_BLEND_1WIN_0

Blend Control for this window area where it does not overlap with other windows.

Offset: 710h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	C_BLEND_WEIGHT1_1WIN: Window blend weight 1 for color key disabled or color key enabled with key matched areas. This is used only for alpha weight with 1-bit alpha and alpha value of 1.
15:8	X	C_BLEND_WEIGHT0_1WIN: Window blend weight 0 for color key disabled or color key enabled with key matched areas. For alpha weight, this is used for 1-bit alpha with value of 0.
2	X	C_BLEND_CONTROL_1WIN: Window blend control in area where it does not overlap with other windows and either color key disabled or color key enabled with key matched. 0 = Fix weight using the window blend weight 0 if color key disabled or color key 0 enabled with key matched, or using the window blend weight 1 if color key 1 enabled with key matched. 1 = Alpha weight using the alpha value. This is only valid if the window color format includes an alpha value. For 1-bit alpha value, if the alpha is 0 window blend weight 0 is used and if alpha is 1, window blend weight 1 is used. 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT
1:0	X	C_CKEY_ENABLE_1WIN: Window color key enable 0 = Color Key 0 and 1 Disable 1 = Color Key 0 Enable 2 = Color Key 1 Enable 3 = Color Key 0 and 1 Enable 0 = NOKEY 1 = CKEY0 2 = CKEY1 3 = CKEY01

29.14.44 DC_WIN_C_BLEND_2WIN_A_0

Blend Control for this window area that overlaps with window A only.

Offset: 711h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	C_BLEND_WEIGHT1_2WIN_A: Window blend weight 1 for color key disabled or color key enabled with key matched areas. This is used only for alpha weight with 1-bit alpha and alpha value of 1.
15:8	X	C_BLEND_WEIGHT0_2WIN_A: Window blend weight 0 for color key disabled or color key enabled with key matched areas. For alpha weight, this is used for 1-bit alpha with value of 0.
3:2	X	C_BLEND_CONTROL_2WIN_A: Window blend control in area where either color key disabled or color key enabled with key matched. 0 = Fix weight using the window blend weight 0 if color key disabled or color key 0 enabled with key matched, or using the window blend weight 1 if color key 1 enabled with key matched. 1 = Alpha weight using the alpha value. This is only valid if the window color format includes an alpha value. For 1-bit alpha value, if the alpha is 0 window blend weight 0 is used and if alpha is 1, window blend weight 1 is used. 2 = Dependent weight, in this case, the window blend weight is 1 - the weights of the other window that overlaps with this window. Only 1 of the overlapping windows may have this setting. 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT 2 = DEPENDENT_WEIGHT
1:0	X	C_CKEY_ENABLE_2WIN_A: Window color key enable 0 = Color Key 0 and 1 Disable 1 = Color Key 0 Enable 2 = Color Key 1 Enable 3 = Color Key 0 and 1 Enable 0 = NOKEY 1 = CKEY0 2 = CKEY1 3 = CKEY01

29.14.45 DC_WIN_C_BLEND_2WIN_B_0

Blend Control for this window area that overlaps with window B only.

Offset: 712h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	C_BLEND_WEIGHT1_2WIN_B: Window blend weight 1 for color key disabled or color key enabled with key matched areas. This is used only for alpha weight with 1-bit alpha and alpha value of 1.
15:8	X	C_BLEND_WEIGHT0_2WIN_B: Window blend weight 0 for color key disabled or color key enabled with key matched areas. For alpha weight, this is used for 1-bit alpha with value of 0.
3:2	X	C_BLEND_CONTROL_2WIN_B: Window blend control in area where either color key disabled or color key enabled with key matched. 0 = Fix weight using the window blend weight 0 if color key disabled or color key 0 enabled with key matched, or using the window blend weight 1 if color key 1 enabled with key matched. 1 = Alpha weight using the alpha value. This is only valid if the window color format includes an alpha value. For 1-bit alpha value, if the alpha is 0 window blend weight 0 is used and if alpha is 1, window blend weight 1 is used. 2 = Dependent weight, in this case, the window blend weight is 1 - the weights of the other window that overlaps with this window. Only 1 of the overlapping windows may have this setting. 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT 2 = DEPENDENT_WEIGHT
1:0	X	C_CKEY_ENABLE_2WIN_B: Window color key enable 0 = Color Key 0 and 1 Disable 1 = Color Key 0 Enable 2 = Color Key 1 Enable 3 = Color Key 0 and 1 Enable 0 = NOKEY

Bit	Reset	Description
		1 = CKEY0 2 = CKEY1 3 = CKEY01

29.14.46 DC_WIN_C_BLEND_3WIN_AB_0

Blend Control for this window area that overlaps with windows A & B only.

Offset: 713h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	C_BLEND_WEIGHT1_3WIN_AB: Window blend weight 1 for color key disabled or color key enabled with key matched areas. This is used only for alpha weight with 1-bit alpha and alpha value of 1.
15:8	X	C_BLEND_WEIGHT0_3WIN_AB: Window blend weight 0 for color key disabled or color key enabled with key matched areas. For alpha weight, this is used for 1-bit alpha with value of 0.
3:2	X	C_BLEND_CONTROL_3WIN_AB: Window blend control in area where either color key disabled or color key enabled with key matched. 0 = Fix weight using the window blend weight 0 if color key disabled or color key 0 enabled with key matched, or using the window blend weight 1 if color key 1 enabled with key matched. 1 = Alpha weight using the alpha value. This is only valid if the window color format includes an alpha value. For 1-bit alpha value, if the alpha is 0 window blend weight 0 is used and if alpha is 1, window blend weight 1 is used. 2 = Dependent weight, in this case, the window blend weight is 1 - the weights of the other window that overlaps with this window. Only 1 of the overlapping windows may have this setting. 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT 2 = DEPENDENT_WEIGHT
1:0	X	C_CKEY_ENABLE_3WIN_AB: Window color key enable 0 = Color Key 0 and 1 Disable 1 = Color Key 0 Enable 2 = Color Key 1 Enable 3 = Color Key 0 and 1 Enable 0 = NOKEY 1 = CKEY0 2 = CKEY1 3 = CKEY01

29.14.47 DC_WIN_C_HP_FETCH_CONTROL_0

Window C High Priority Avoidance Fetch Parameters

This register gives extra information to the Memory Controller Client Interface (MCCIF) about the number of memory words that will be fetched per scan line and about the rate at wich those words are consumed. This allows the MCCIF to more accurately arbitrate memory accesses to prevent the use of the High Priority signal.

Use of this signal causes all other client accesses to be blocked in preference to the client asserting HP. Obviously, this is a state which is sometime necessary for Display since it is an isochronous client and MUST be serviced in a timely manner. However, use of this signal should be avoided if possible. These parameters help the MCCIF avoid the over-use of HP.

Offset: 714h | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	0x0	C_FETCH_INFO_ENABLE: Enables the sending of the Window C fetch information. For compatibility with earlier devices, this defaults to DISABLE. 0 = DISABLE : This bit should be enabled only for 12-bpp 1 = ENABLE

Bit	Reset	Description
30:16	X	C_WORDS_PER_LINE: Window C memory fetch words per scan line. This value is in memory fetch words: Multiples of 16 bytes for Tegra 2 Processor Series devices. It is computed as follows: C_WORDS_PER_LINE = (C_SIZE.C_H_SIZE * (bytes per pixel) + 15) >> 4 bytes per pixel is determined by the pixel format.
15:0	X	C_CYCLES_PER_WORD: Window C clock cycles per memory fetch word. The value of this field is essentially a measure of the data consumption rate for window C. It is computed as follows: C_CYCLES_PER_WORD = C_DDA_INCREMENT.C_H_DDA_INCREMENT / (bytes per pixel) Note that the format for this value is a fixed-point fractional value with 8 bits of integer precision and 8 bits of fractional precision. In other words, it is an '8.8' number. For example, if there is no scaling of the input image, the DDA increment will be 4096. With 32-bit RGBA pixels there will be 4 bytes per pixel, so CYCLES_PER_WORD will be ... 4096 / 4 = 1024, or 4.0 expressed in the 8.8 format. Any scaling performed on the pixels will change the rate at which pixels are consumed. Scaling up will increase the value of DDA increment and will therefore increase the number of cycles between memory fetches. Conversely, scaling down will decrease the value of DDA increment and memory fetches will occur more frequently.

29.15 WINBUF_C Registers

The registers under DC_WINBUF are triple-buffered.

29.15.1 DC_WINBUF_C_START_ADDR_0

Window C Start Address

Overview

START_ADDR, BUF_STRIDE, LINE_STRIDE, ADDR_H_OFFSET, ADDR_V_OFFSET combined, specify the starting address of a window buffer in the memory surface. Generally START_ADDR is programmed with the starting address of the memory surface. H/V_OFFSET specifies the address offsets of the pixel at the beginning of the window, with respect to the starting address of the memory surface. (There is an exception to that when memory surface is not well aligned, see 3-ii below.)

Note that "beginning of the window" here means the top-left corner of a window in normal mode, top-right corner in horizontal flipping, bottom-left corner in vertical flipping, and bottom-right in horizontal+vertical flipping.

Starting Address Calculation

These formulae are same for both tiled or linear mode. However, for linear mode, the addresses calculated are real physical addresses, but for tile mode, these addresses will be translated to the real physical addresses by the memory controller client before used.

- When a window is host triggered, starting address of a window is calculated as follows by HW.
  - non-yuv-planar modes

    starting-address = START_ADDR + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET
  - yuv-planar modes

    y-starting-address = START_ADDR + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET

    u-starting-address = START_ADDR_U + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2

    v-starting-address = START_ADDR_V + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2 where denom1/denom2 equal to 1 or 2 depending on the actual planar format, derived natively by HW.

- When a window is non-host triggered, starting address of a window buffer is calculated as below.
  - non-yuv-planar mode

    starting-address = START_ADDR + BUF_STRIDE * buf_index + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET
  - yuv-planar mode:

    y-starting-address = START_ADDR + BUF_STRIDE * buf_index + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET

    u-starting-address = START_ADDR_U + UV_BUF_STRIDE * buf_index + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2

    v-starting-address = START_ADDR_V + UV_BUF_STRIDE * buf_index + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2 where denom1/denom2 equal to 1 or 2 depending on the actual planar format, derived natively by HW.

    buf_index is the index transmitted by the triggering module pointing to the first buffer of a frame.

Programming Restrictions

- For tiled address mode

  Image surface can only aligned to multiples of 256, thus the following restrictions.
  - START_ADDR, START_ADDR_U, START_ADDR_V need to be multiples of 256.
  - BUF_STRIDE, UV_BUF_STRIDE need to be multiples of 256
  - LINE_STRIDE, UV_LINE_STRIDE need to be multiples of 16
  - ADDR_H_OFFSET needs to be multiple of 2 in yuv planar format (the last bit is ignored), but with no restrictions on other color formats.
  - ADDR_V_OFFSET has no restrictions
- For linear address mode

  Image surface can be aligned 2, 4 or 8 bytes, depending on the color formats.

  As an additional restriction for display, START_ADDR, START_ADDR_U and START_ADDR_V need to be multiples of 16. When a surface is not aligned to 16 bytes, program START_ADDR with the memory surface address with its least 4 significant bits zeroed out and add these 4 address bits to the original H_OFFSET. (So the formula in 2-i,ii still hold)
  - For all formats:

    START_ADDR, START_ADDR_U and START_ADDR_V need to be multiples of 16.

    For 16-bpp formats,

    (START_ADDR+H_OFFSET) need to be multiple of 2. The least significant bit of H_OFFSTE is ignored.

    For 32-bpp formats,

    (START_ADDR+H_OFFSTE) needs to be multiple of 4. The least two significant bits of H_OFFSTE are ignored.
  - For yuv planar formats:

    BUF_STRIDE, UV_BUF_STRIDE:

    BUF_STRIDE[2:1]=UV_BUF_STRIDE[1:0]

    or as a stricter constraint: BUF_STRIDE be multiple of 8, UV_BUF_STRIDE be multiple of 4.

    LINE_STRIDE, UV_LINE_STRIDE:

    LINE_STRIDE and UV_LINE_STRIDE need to be at least 16.

    LINE_STRIDE needs to be multiple of 8, UV_LINE_STRIDE needs to be multiple of 4.

    ADDR_H_OFFSET:

Needs to be multiple of 2. If needs to point to odd pixel position, program ADDR_H_OFFSET to be the previous position (or the next position if H-flipped) and program H_INITIAL_DDA bit 12 to 1.

ADDR_V_OFFSET:

Needs to be multiple of 2. If needs to point to odd line number, program ADDR_V_OFFSTE to be the previous line number (or next line number if V-flipped) and program V_INITIAL_DDA bit 12 to 1.

- Memory allocation for non-host triggered case:

  When a window buffer is not controlled by host (software) then a frame may be stored in multiple buffers. In this case, the buffers must be contiguous in the memory because display will use the same luma or chroma line strides for all lines in the frame.

  Also buffer wraparound must not occur in the middle of the displayed part of the frame. The controlling module will send frame start and frame end indicators (flags) to display module to indicate the beginning and end of frame. Buffer start address is latched when frame start flag is active but the actual buffer start address is not switched until frame end flag is active. In the case where one buffer correspond to one frame then frame start and frame end flag are active everytime a buffer index is sent.

Offset: 800h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	C_START_ADDR: Window C Start Address This is a byte address. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSB are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies start address for the Y plane.

29.15.2 DC_WINBUF_C_START_ADDR_NS_0

Window C Shadowed Start Address

Offset: 801h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	C_START_ADDR_NS: Window C Shadowed Start Address This is ARM set shadow of Start Address.

29.15.3 DC_WINBUF_C_START_ADDR_U_0

Window C Start Address for U plane

Offset: 802h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	C_START_ADDR_U: Window C Start Address for U plane This is a byte address.

29.15.4 DC_WINBUF_C_START_ADDR_U_NS_0

Window C Shadowed Start Address for U plane

Offset: 803h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	C_START_ADDR_U_NS: Window C Shadowed Start Address for U plane This is ARM set shadow register of U start address

29.15.5 DC_WINBUF_C_START_ADDR_V_0

Window C Start Address for V plane

Offset: 804h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	C_START_ADDR_V: Window C Start Address for V plane This is a byte address.

29.15.6 DC_WINBUF_C_START_ADDR_V_NS_0

Window C Shadowed Start Address for V plane

Offset: 805h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	C_START_ADDR_V_NS: Window C Shadowed Start Address for V plane This is ARM set shadow register of U start address

29.15.7 DC_WINBUF_C_ADDR_H_OFFSET_0

Window C Horizontal address offset

Offset: 806h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	C_ADDR_H_OFFSET: Window C Horizontal address offset This is a byte address. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSB are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies horizontal offset of Y plane. The horizontal offsets of U/V plane is derived by HW.

29.15.8 DC_WINBUF_C_ADDR_H_OFFSET_NS_0

Window C Shadowed Horizontal address offset

Offset: 807h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	C_ADDR_H_OFFSET_NS: Window C Shadowed Horizontal address offset This is ARM set shadow of ADDR_H_OFFSET

29.15.9 DC_WINBUF_C_ADDR_V_OFFSET_0

Window C Vertical address offset

Offset: 808h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	C_ADDR_V_OFFSET: Window C Vertical address offset This is a byte address. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSB are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies vertical offset of Y plane. The vertical offsets of U/V plane is derived by HW.

29.15.10 DC_WINBUF_C_ADDR_V_OFFSET_NS_0

Window C Shadowed Vertical address offset

Offset: 809h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	C_ADDR_V_OFFSET_NS: Window C Shadowed Vertical address offset This is ARM set shadow of ADDR_V_OFFSET

29.15.11 DC_WINBUF_C_UFLOW_STATUS_0

Window C FIFO Underflow Status Register

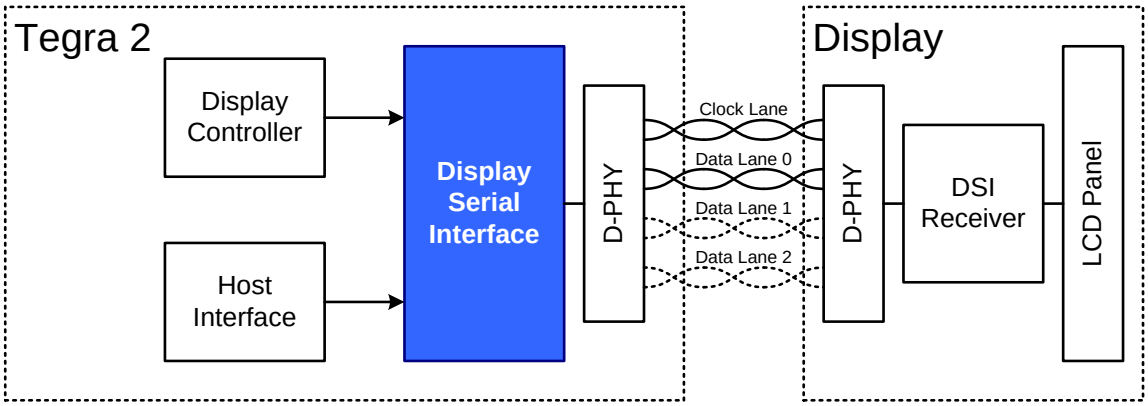
Offset: 80ah | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
30	X	COUNT_OFLOW: Flag bit that indicates that the underflow event counter has overflowed. There were too many events. If COUNT_OFLOW is set, UFLOW_COUNT is meaningless. Cleared on write.
23:0	X	UFLOW_COUNT: Underflow count. This field indicates the number of contiguous groups of output pixels for which there was no data in the FIFO. For example, if the valid from the FIFO is low for 10 consecutive cycles and then goes high, the counter will increment by one. Reset to zero on write.

30.0 DISPLAY SERIAL INTERFACE (MIPI-DSI)

The Display Serial Interface (DSI) is a MIPI standard serial bit-stream, intended to provide a low pin count interface to a display panel. DSI reduces both package pin-count and I/O power consumption compared to parallel solutions. It transfers pixel data from either one of the display controllers (internal to the Tegra® 2 Processor) 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 93.

Figure 93 System block diagram



DSI is a replacement for the MIPI DPI and DBI standards, and follows the use cases of these interfaces. From a data transport point of view, MIPI DPI is an interface similar to a traditional raster-based isochronous display interface, and DBI is an asynchronous packet based transfer mechanism. DSI behaves like MIPI DPI when in “Video Mode”, and implements a “Command Mode” to handle the DBI interface. Any implementation must implement both these modes of operation.

30.1 Clocking

- Host Clock = 166 MHz {Max frequency}
- Pixel Clock = 2.49 to 91.0 MHz for all modes
- Application/Lane Management Byte Clock = function of lanes, pixel clock, pixel depth. 10 to 125 MHz
- Fixed LP receive clock = 72 MHz
- Lane Bit clock = 40 to 500 MHz differential clock, DDR data (80 to 1000 bits/sec)

More details on pixel clock/byte_clock selection is explained in the next section.

30.1.1 Pixel Clock / DSI Byte Clock Ratios

When running the DSI interface in one of the two non-burst video modes, the relationship between the DSI D-PHY bit-rate clock, the DSI byte-rate lane clock and the display controller pixel clock must be exact. This will ensure that precise raster timing information is conveyed to the display peripheral.

There is always a fixed ratio of 8:1 between the D-PHY bit clock and the lane management layer byte clock since there are 8 bits in each byte that are serialized. Therefore, only relationships between pixel clock and byte clock and between pixel clock and bit clock will be discussed.

The relationship between pixel clock and byte clock is shown in Table 103. As can be seen, for some modes, these ratios are far from being simple powers of 2.

Table 103. Pixel Clock: Byte Clock for Various Modes.

Lanes	Pixel Data Format			
	16 bpp	18 bpp (packed)	18 bpp	24 bpp
1	1:2	4:9	1:3	1:3
2	1:1	8:9	2:3	2:3

Since the pixel clock is effectively derived from the D-PHY bit clock, it is perhaps more instructive to look at the number by which you must divide the bit clock to get the correct pixel clock. This information is shown in Table 104. Here, the numbers look reasonable except for the problematic cases of 16bpp over a 3-lane link and 18 bpp (packed) over a 4 lane link. For these two situations, you cannot simply divide the bit rate clock by some integer to arrive at the pixel clock. Moreover, if you need a 50:50 duty cycle for the pixel clock, then you will need to toggle the pixel clock every N/2 bit clock cycles, where N is the number in 2. However, if you do this, 18 bpp (packed) over 2 lanes also becomes a problem.

Table 104. Value to Divide Bit Clock by to Get Pixel Clock

Lanes	Pixel Data Format			
	16 bpp	18 bpp (packed)	18 bpp	24 bpp
1	16	18	24	24
2	8	9	12	12
3	16 / 3	6	8	8
4	4	9 / 2	6	6

To resolve these issues, derive the display controller pixel clock from a separate PLL and then lock that PLL to the DSI D-PHY PLL using a phase comparator and the appropriate divide ratios obtained from the tables. In the straightforward case where pixel clock is derived from PLLD, programming pixel clock divider using Table 104 above and PLLD programming will suffice clocking requirements. PLLD is covered in section 30.6.1 below.

30.2 Operating Modes

30.2.1 Video Mode

Communication with the peripheral is by isochronous data transfer similar to a typical video interface. There is no Bus Turn Around permitted in Video Mode, though there is a mandatory period of LP operation at least once per frame. There are three different sub-modes of Video Mode. These are outlined below:

30.2.1.1 Non-Burst Mode with Start and End

In this mode, the DSI must match the timing of a traditional video raster as closely as possible. This means all information about the start and end of parameters like vertical and horizontal sync, vertical and horizontal sync, front and back porches must be conveyed to the receiving peripherals via the timing of the transmission of the High-Speed sync packets.

30.2.1.2 Non-Burst Mode

This is like Non-Burst Mode with Start and End, except that the Sync-Active and Sync-End packets are not sent. Pixels must still be delivered at the correct rate.

30.2.1.3 Burst Mode

Only the timing of Sync-Start packets is required to be accurately conveyed in this mode. The data rate of the pixel data is arbitrary and is typically higher than the pixel rate of the peripheral. This allows time to transmit other information during the periods where no more pixels are needed.

30.2.2 Command Mode

Communication with the peripheral is by asynchronously timed and variable length packets. The packets may contain video data or control data. In Command Mode, return (read) data can be requested from the peripheral. The DSI will issue a Bus Turn Around (BTA) request and relinquish control of the bus to the peripheral.

30.3 Display Controller Interface

The interface between the display controller and the DSI unit consists of the following elements:

- 24-bit wide pixel bus with an associated valid signal. The valid will only be high during the active portion of a video line.
- 1-bit Horizontal Sync (HS) signal that will toggle once per line.
- 3-bit Lin-Type code. These bits should remain stable for at least 2 pixel clock periods prior to the Horizontal Sync signal changing state. This is to ensure that on the DSI side, the state of these 3 bits is correctly captured.

The DSI unit uses these signals in the following manner: When the display controller toggles the HS signal, the DSI captures the state of the line type code. This information is then used to look up a pre-programmed packet sequence that corresponds to that line type. There are several different line types to select from. Once the packet sequence has been selected, the DSI sends these packets into its internal packet FIFO. When a packet that requires data from the display controller is encountered, the DSI pauses the packet writing operation, until data arrives from the display controller on the pixel bus. When the data arrives – indicated by the assertion of the valid signals – the data is written into the FIFO. This process continues until all the packets for that line type have been written.

Separately, when the HS signal arrives, a counter is started. When the counter reaches a pre-programmed value, the logic on the read side of the DSI internal FIFO is instructed to start pulling packets out of the FIFO and transmit them over the DSI link to the display device.

30.4 Host Interface

Not all displays can accept a raster-based packet stream. For some low-bandwidth displays with small pixel and line dimensions, it is more efficient to have a frame store in the display device itself and to only update the contents of the frame store with occasional packet sequences. For these types of display, there is a SW accessible interface for sending command packets using DSI command mode operation. This mode should also be used to access control and status registers within the display device.

The Host interface consists of a small FIFO for holding packet information and a special register for writing data into the FIFO. In this way, several packets may be queued up by SW and sent in one HS burst. This is more efficient that sending the packets one at a time.

When very long sequences of packets are required to be sent – for example, when the host interface is emulating video mode, or where a large amount of data needs to be sent to the display device in one packet – it is possible for the host to use the large video line-store FIFO.

30.5 FIFO Buffers

30.5.1 Video Mode Operation

30.5.1.1 Writing Data

The following packets are written into the large data FIFO based on the various triggering events described.

Table 105 FIFO Trigger Events

Trigger Event	Packet
Leading edge of Vertical Sync.	V Sync Start
Trailing edge of Vertical Sync	V Sync End
Leading edge of Horizontal Sync, *unless* simultaneous with leading or trailing edge of vertical sync	H Sync Start
Trailing edge of Horizontal Sync	H Sync End
Immediately following VS, HS, VE or HE packets. Packet Word Count determined by display controller timing parameters.	Blanking packet
Immediately following blanking packet	Pixel Stream packet (16, 18 or 24 bits/pixel)

30.5.1.2 Reading Data

The initiation of the reading of the FIFO is triggered by a delayed version of the horizontal sync pulse from the display controller. The delay should be sufficient that the FIFO will not completely drain as reading of the FIFO continues. However, the delay should not be so long as to cause the FIFO to overflow with data.

30.5.2 Command Mode Operation

30.5.2.1 Writing Data from the Display Controller

Upon receipt of the leading edge of Vertical Sync, the display sends whatever DCS command packets are required to initialize the display to receive pixels. Then, as pixels start to arrive from the display controller, a DCS Long Write packet is sent – one per line – which contains the pixel data for that line. No synchronization or blanking packets should be sent. Data packets continue to be sent until the end of the active period. Like video mode packets, there is no need to calculate ECC or CRC words, as this is performed by the hardware on the other side of the data FIFO.

The Packet Sequence registers behave in the same manner as they do for Video Mode packet generation with the following exceptions:

- For Long packets, a DCS command ID byte must be inserted as the first byte of the data payload of the packet, ahead of the pixel data. The packet header Word Count must be 1 more than the number of bytes in the pixel data to take into account this extra byte, but it is the responsibility of SW to make sure the packet length is programmed correctly.
- For Line Type 4, any data contained in the Host Data FIFO should be appended to the end of any packets defined in the Packet Sequence for Line Type 4. In this way, the Host may send DCS commands to the Display Peripheral while the Display Controller is sending pixel information to the Display Peripheral.

30.5.2.2 Writing Data from the Host

Host should choose which data FIFO – small or large – to use prior to sending packet information. The large data FIFO is only available if the display controller is not currently using it. The software is responsible for constructing whatever packets are required for the desired operation and should be written into the FIFO via the host. There is no need for the software to compute ECC or CRC words, as this is performed by the hardware on the other side of the data FIFO.

30.5.2.3 Reading Data

The initiation of the reading of data from the FIFO is triggered by a FIFO threshold. Once the threshold is reached, the FIFO starts to be drained by the D-PHY and the packets are sent to the display. The threshold should be set such that there is no danger of either FIFO overflow from excessive data from the Host or display controller, nor FIFO underflow caused by a lack of sufficient data in the FIFO.

30.5.3 FIFO Sizing

FIFO Threshold Video Non-bust mode is calculated as:

Threshold = (Length x OutRate) / (OutRate + InRate)

Where,

- InRate = pixClk
- OutRate = (ByteClk x NumOfLanes) / BytePerPix
- Length = input frame width
- Threshold = minimum fifo depth to start transfer data to CIL

Table 106 Output Rates and Thresholds for Width=800, ByteClk=128, InRate=44.

		Number of Lanes			
Bytes per pixel		1	2	3	4
2	Out Rate	64	128	192	256
2	Threshold	474	595	650	682
2.25	Out Rate	56	113	170	227
2.25	Threshold	448	575	635	670
3	Out Rate	42	85	128	170
3	Threshold	390	527	595	635

Table 107 Output Rates and Thresholds for Width=1024, ByteClk=128 InRate=44.

		Number of Lanes			
Bytes per pixel		1	2	3	4
2	Out Rate	64	128	192	256
2	Threshold	606	762	833	873
2.25	Out Rate	56	113	170	227
2.25	Threshold	573	737	813	857
3	Out Rate	42	85	128	170
3	Threshold	500	674	762	813

30.6 Programming Guidelines

The packet timing diagrams in this section use the following key:

Figure 94 Video Mode Timing Diagram Key

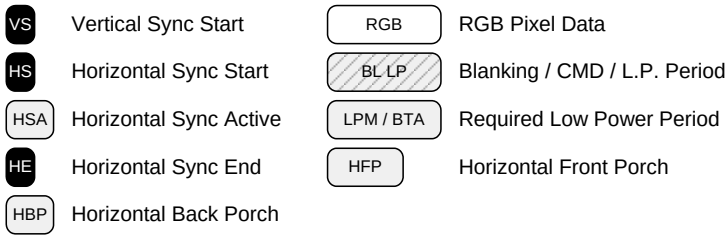


Table 108 Timing Diagram parameter list

Parameter	Description
tHBP	Horizontal Back Porch period
tHACT	Horizontal Active period
tVBP	Vertical Back Porch period
tVACT	Vertical Active period
tVFP	Vertical Front Porch period
tL	Total line period

Table 109 DSI Register Overview

Register	Description
DSI_RD_DATA	BTA read-return FIFO output
DSI_WR_DATA	Host Transmit FIFO input
DSI_POWER_CONTROL	
INT_ENABLE	Interrupt Enables
INT_STATUS	Interrupt Status
INT_MASK	Interrupt Mask
HOST_DSI_CONTROL	Host-specific DSI controls
DSI_CONTROL	General and Video DSI controls
DSI_SOL_DELAY	Start-Of-Line delay register
DSI_MAX_THRESHOLD	FIFO transmit start threshold
DSI_TRIGGER	Manual transmit trigger register
DSI_TX_CRC	Verification transmitted data CRC
DSI_INIT_SEQ_CONTROL	Initialization sequence control
DSI_INIT_SEQ_DATA_0	Initialization data 0
DSI_INIT_SEQ_DATA_1	Initialization data 1
DSI_INIT_SEQ_DATA_2	Initialization data 2
DSI_INIT_SEQ_DATA_3	Initialization data 3
DSI_PKT_SEQ_[0..5]_LO	Packet Sequence [0..5] (low)
DSI_PKT_SEQ_[0..5]_HI	Packet Sequence [0..5] (high)
DSI_DCS_CMDS	DCS commands for Line Type 3 and 5
DSI_PKT_LEN_0_1	Packet Lengths for packets 0 and 1
DSI_PKT_LEN_2_3	Packet Lengths for packets 2 and 3

Register	Description
DSI_PKT_LEN_4_5	Packet Lengths for packets 4 and 5
DSI_PKT_LEN_6_7	Packet Lengths for packets 6 and 7
DSI_PHY_TIMING_0	Low-level D-PHY timing control
DSI_PHY_TIMING_1	Low-level D-PHY timing control
DSI_BTA_TIMING	Low-level D-PHY timing control
DSI_TIMEOUT_0	HTX_TO and LRXH_TO durations
DSI_TIMEOUT_1	TA_TO and PR_TO durations
DSI_TO_TALLY	Count of how many TOs have occurred

30.6.1 Initialization

The start-up and shut-down procedures vary depending on what mode of operation is required. Examples of programming sequences for the main modes are given below.

30.6.1.1 PLLD Programming

The DSI PLL is called PLLD in the Tegra 2 Processor. Just like the other PLLs in the system, PLLD has an M, N and P register for programming the various divide ratios for the clock. When used to drive Display pixel clock and DSI byte clock, the P divider is not used. Instead, there is a second divider that is used specifically for obtaining the Display pixel clock. The division ratio is labeled as D in the code below, but the correct name of the register should be used when actually programming the PLL.

```
//**** Environment ... ****

const float Fomin = 80.0;   // Min. oscillator frequency (MHz)
const float Fomax = 1000.0; // Max. oscillator frequency (MHz)
const float Fcmin = 1.0;    // Min. PLL phase comparison frequency (MHz)
const float Fcmax = 6.5;    // Max. PLL phase comparison frequency (MHz)

// MIPI Pixel formats ...
typedef enum { BIT16P, BIT18P, BIT18NP, BIT24P } pixel fmt;

//**** PLL Programming Information ... ****

int   M;         // Value by which the Reference clock is divided
                 // to get the comparison frequency
int   N;         // Value by which the Oscillator clock is divided
                 // to get the comparison frequency
int   D;         // Ratio between D-PHY bit clock and pixel clock

float Fosc;      // PLL oscillator frequency, in MHz   (information only)
float Fcomp;     // Phase comparator frequency, in MHz (information only)


//**** Function that computes the values of M, N and D ... ****

bool calc M N D(float Fref, float Fpixel, float tolerance,
                pixel fmt fmt, int lanes)

/******************
  Fref      : Input Reference frequency - in MHz
  Fpixel    : Desired output pixel clock frequency
  tolerance : Fractional error allowed in the output frequency
```

```
  fmt       : Pixel format. One of BIT16P, BIT18P, BIT18NP and BIT24P
  lanes     : Number of DSI lanes in use
  Function returns “true” on success and “false” otherwise
********************/

{
  float Nfrac;
  bool  done = false;

  // Compute the value of D ...

  switch (fmt) {
    case BIT16P  : D = 16; break;
    case BIT18P  : D = 18; break;
    case BIT18NP :
    case BIT24P  : D = 24; break;
    }

  D = D / lanes;
  Fosc = Fpixel * clk ratio; // Calculate the Oscillator Frequency

  // Compute the values of M and N ...

  if ((Fosc > Fomin) && (Fosc < Fomax)) {
   // If the oscillator frequency is "legal" ...
    M = (int)(0.99 + Fref / Fcmax);
    do {
      // For each value of M, compute the comparison frequency
      Fcomp = Fref / (float)(M);
      // Now compute the exact value of N ...
      Nfrac = Fosc / Fcomp;
      // Strip off the integer and fractional parts ...
      N     = (int)(Nfrac);
      Nfrac = Nfrac - (float)(N);
      if (N > 0) {
        if ((Nfrac / N) < tolerance)
          // If the error is less than the tolerance, then we're done
          done = true;
        else
          // Too much error, so try dividing the reference some more ...
          M++;
        }
      else // N too small, so divide the reference by some more ...
        M++;
      } while (!done && (M < 32) && (Fcomp > Fcmin));
    }

  return (done);
}
```

30.6.1.2 Video Mode Start-up

All 3 video modes – Burst, Non-Burst and Non-Burst with Sync Ends – essentially have the same start-up sequence, with a slight variation between Burst and Non-Burst modes:

- Set up the clocks. This involves configuring and enabling the DSI PLL (PLLD). For Non-Burst and Non-Burst with Sync End modes, the Display Controller must also be programmed to take its pixel clock from the DSI PLL. For Burst Mode, the Display Controller can take its pixel clock either from the DSI PLL or from another clock source. Program the PLLD registers with the correct OSC_FREQ and program the PLLD_BASE register. In addition, PLLP must be running in order to provide the LP receive clock. Program the PLLP registers with the correct OSC_FREQ and program the PLLP_BASE register.
- Depending on the sub-mode, the various Packet Sequence registers and Packet Length registers must be programmed.
- Set the `VID_TX_TRIG_SRC` field in the `DSI_CONTROL` register to `SOL`. Select which Display Controller will source the video data.
- Configure PHY timing registers and timeout registers
- Enable DSI
- Program the Display Controller to produce the raster size required.
- Enable the Display Controller.

When the Display Controller starts sending SOL and data, the DSI automatically starts creating and transmitting packets to the Display Device.

30.6.1.3 Command Mode start-up

Set up the clocks. Program and enable the DSI PLL (PLLD). In addition, PLLP must be running in order to provide the LP receive clock. Program the PLLP registers with the correct OSC_FREQ and program the PLLP_BASE register. Unlike the Video Modes, the selection of PLLD output clock frequency is essentially arbitrary. The suitability of the clock frequency should be based on the expected data throughput and the requirements of the particular Display Device. Program the PLLD registers with the correct OSC_FREQ and program the PLLD_BASE register. In addition, PLLP must be running in order to provide the LP receive clock. Program the PLLP registers with the correct OSC_FREQ and program the PLLP_BASE register.

- Set the `HOST_TX_TRIG_SRC` field in the `HOST_DSI_CONTROL` registers to `IMMEDIATE`. Set any other state required.
- Configure PHY timing registers and timeout registers
- Enable DSI.

The DSI should now be ready to accept Command Packets written to the `DSI_WR_DATA` register.

30.6.2 Video Mode Programming

Packet sequences for video mode are somewhat complex and there is on-going debate within the MIPI DSI workgroup regarding the precise sequences and whether or not certain packets are optional. In order to make the hardware flexible enough to cope with all current video mode sequences and any possible future sequences or sequence variations, a more-or-less open-ended programming model has been adopted.

The programming model is as follows:

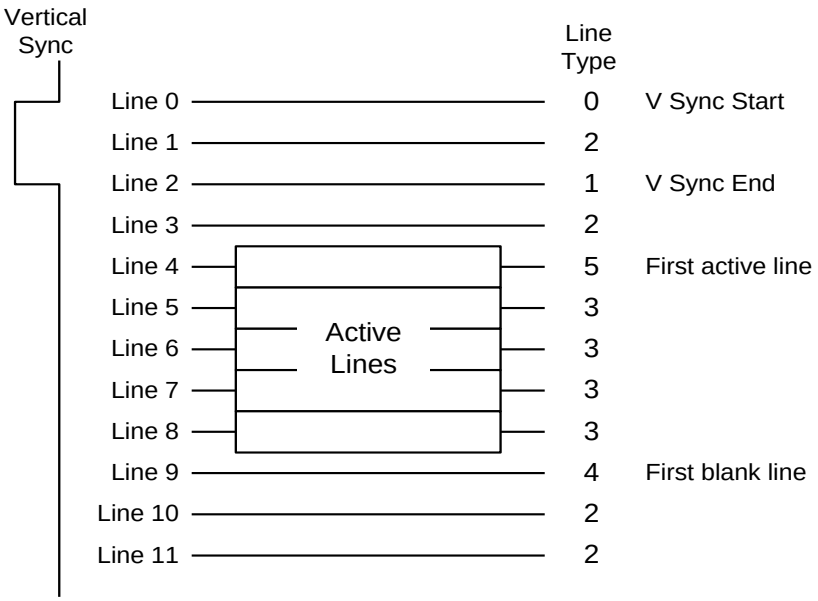
Each line of video output will have associated with it a “line type”. This line type is automatically generated by the display controller HW according to the following table:

Table 110 Line type descriptions

Line Type	Description	Typical raster position
0	Blank line starting with VS	First line of the raster
1	Blank line starting with VE	Line corresponding to V sync end.
2	Blank line starting with HS	Vertical blanking line
3	Active line starting with HS	Active line
4	First blank line after active	First blank line after active
5	First active line	First active line
6, 7	Reserved	

Figure 95 illustrates a typical raster showing the relationship between various events in the raster and the line types.

Figure 95 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 111 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 112 shows how all six packet descriptors fit into the pair of 32-bit packet sequence registers.

Table 112 Packet Sequence 0 registers

Register	Field	Bits	Description
DSI_PKT_SEQ_0_LO	PKT_00_SIZE	2:0	Packet 0 Size
	PKT_00_ID	8:3	Packet 0 ID
	PKT_00_EN	9	Packet 0 Enable
	PKT_01_SIZE	12:10	Packet 1 Size
	PKT_01_ID	18:13	Packet 1 ID
	PKT_01_EN	19	Packet 1 Enable
	PKT_02_SIZE	22:20	Packet 2 Size
	PKT_02_ID	28:23	Packet 2 ID
	PKT_02_EN	29	Packet 2 Enable
	SEQ_0_FORCE_LP	30	End in LP state
DSI_PKT_SEQ_0_HI	PKT_03_SIZE	2:0	Packet 3 Size
	PKT_03_ID	8:3	Packet 3 ID
	PKT_03_EN	9	Packet 3 Enable
	PKT_04_SIZE	12:10	Packet 4 Size
	PKT_04_ID	18:13	Packet 4 ID
	PKT_04_EN	19	Packet 4 Enable
	PKT_05_SIZE	22:20	Packet 5 Size
	PKT_05_ID	28:23	Packet 5 ID
	PKT_05_EN	29	Packet 5 Enable

Finally, the SIZE field in the packet descriptor is used as a pointer to a 16-bit "*packet length*" field in the packet length registers. The reason an indirect pointer is used, rather than storing the packet length directly in the packet descriptor is that the packet lengths are physically large – 16 bits – but there is a lot of re-use. There are actually only a few different lengths used in a typical raster layout. Thus, it is more efficient to hold the lengths in separate length registers and then to simply refer to them with a short pointer. This allows the storing of 36 packet descriptors in only 6 pairs of packet sequence registers.

It should be noted that one of the packet length registers is reserved for short packets. If a `SIZE` field in the packet descriptor is programmed to 0, this implies this packet is a short packet. Short packets are fixed in length to 4 bytes including the packet header byte and ECC byte. In the context of video mode, they are essentially reserved for timing packets. All other packet length registers can be used to determine the length of any associated long packet.

The `SEQ_0_FORCE_LP` bit is used to determine if the link should be placed in the LP state at the end of the packet transmission. In Burst Mode operation, the link always drops back into LP state at the end of the HS packet transmission on a line. However, for Non-Burst Modes, the HS transmission may continue to the next line. It is important the that the HW state machine that generates packets not attempt to go to the LP state, but should instead prepare for the next sequence of packets associated with the next video line and keep the line in the HS transmission state.

The packet length registers are shown in Table 113. The packet size pointer in the packet descriptor can point to any of the available length registers – with the one exception of short packets, whose `SIZE` field must point to length register 0. This open-ended possibility is very powerful, but may lead to confusion. It is therefore recommended that the suggested packet length assignment for various length registers is followed. This consistency will hopefully reduce confusion when debugging packet descriptors and packet sequences.

Table 113 Packet Length registers and assignments

Register	Field	Suggested Assignment
DSI_PKT_LEN_0_1	LENGTH_0	Must be used for short packets
	LENGTH_1	H sync active length (HSA)

Register	Field	Suggested Assignment
DSI_PKT_LEN_2_3	LENGTH_2	H back porch length (HBP)
	LENGTH_3	H active length (ACTIVE)
DSI_PKT_LEN_4_5	LENGTH_4	H front porch length (HFP)
	LENGTH_5	- reserved -
DSI_PKT_LEN_6_7	LENGTH_6	- reserved -
	LENGTH_7	- reserved -

When programming the length registers, it should be remembered that these are byte counts, not pixel counts. Therefore, the DSI pixel format, number of DSI lanes in use and the finite (non-zero) size of the packet header and CRC words should be taken into account when programming these values – especially for blanking packets since all of these parameters affect the number of bytes that should be used to emulate a specific blanking time. The equations required to determine the byte count in the packets is given in the examples that follow. The identification of the length given in the tables and equations has been included in braces in Table 113 so that the values in the examples can be easily tied to the registers.

30.6.3 SOL Delay Programming

In order to ensure that the pixel FIFO from display to DSI does not underflow when in video mode, it is necessary to delay the start of packet generation by the DSI, by a fixed amount from the arrival of the SOL signal from display. This is especially true of Burst-Mode, since the DSI byte clock can be very much faster than display pixel clock. If no delay is applied, the DSI will very quickly consume all the pixels in the FIFO and the FIFO will underflow. The `SOL_DELAY` registers is used to set this delay.

30.6.3.1 None-Burst Mode

In non-burst mode, the rate at which DSI consumes pixels is the same as the rate at which the display module produces them, so it is merely sufficient to ensure that enough pixels have been fed into the FIFO to overcome the internal latency. This internal latency is on the order of 8 pixel clock cycles. However, `SOL_DELAY` is programmed in DSI byte clocks, so the pixel format and the number of lanes being used should be taken into account when programming this value. Please contact your NVIDIA FAE for details on the relationship between display pixel clock and byte clock. These ratios are then used to determine the value of `SOL_DELAY` as follows:

$$SOL\_DELAY = 8 * F_{DSI} / F_{pixel}$$

Where F_{DSI} and F_{pixel} are the DSI byte and pixel clocks, respectively.

Alternatively, a fixed value of `SOL_DELAY` that will work for all pixel formats and numbers of lanes can be programming by taking the worst-case ratio of 1:3 (pixel:byte) clock and multiplying by 8 to give `SOL_DELAY` = 24.

30.6.3.2 Burst Mode

Burst mode is much more complex because not only must the pixel format, number of lanes and the DSI / pixel clock ratio be taken into account, but also the horizontal back porch time (including H sync) and the H active time. So, for Burst Mode:

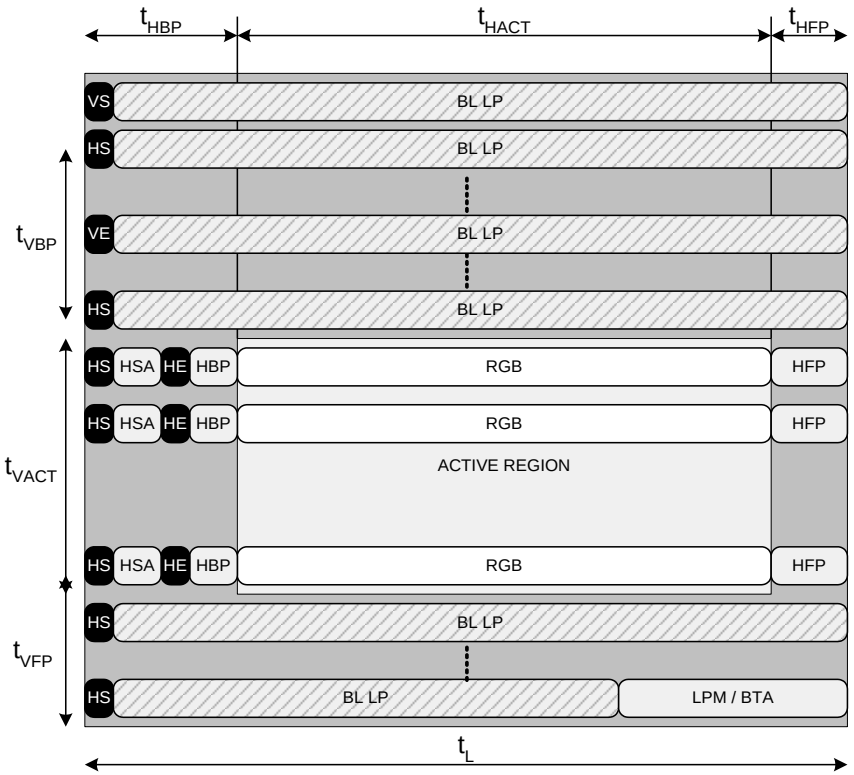
$$SOL\_DELAY = ((T_{HBP} + T_{active}) * F_{DSI} / F_{pixel}) – (T_{active} * F_{DSI\_NB} / F_{pixel\_NB})$$

Where the ratio $F_{DSI\_NB} / F_{pixel\_NB}$ is determined by the clock ratio tables for Non-Burst mode, according to the pixel format used and the number of active lanes.

30.6.4 Non-Burst Mode with Start and End

In this mode, an attempt is made to accurately convey traditional raster synchronization information across the link to the peripheral. This is achieved by sending both start and end sync packets.

Figure 96 Timing Diagram for Video Non-Burst Mode with Start and End



During the Vertical Active period, all packets should be concatenated into a single HS transmission for the duration of the Vertical Active period. There is some debate going on as to whether the HFP blank packet is required, or even desirable. This may get replaced by a period of Blanking / Low Power mode.

The detail of how a single line is constructed from multiple packets is shown in the figure below. In addition, it is necessary for the packet constructor on the display clock side of the display / DSI boundary FIFO to calculate the packet size for four different packets on the line.

It is also suggested that there is a running-disparity adjustment made to the packet sizes so that the average pixel rate on the DSI side matches the display output rate.

Figure 97 Non-Burst Mode with Start and End Packet Timing Detail

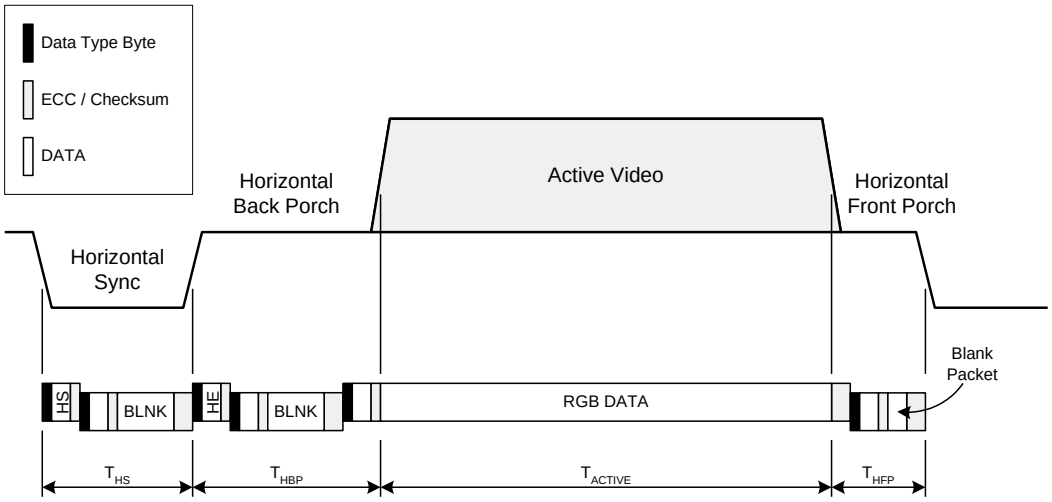


Table 114 Payload Size Table - Non-Burst Mode with Start and End

Packet	Payload Size (Bytes)
HSA	(T_{HS} * B) – 10
HBP	(T_{HBP} * B) – 14
ACTIVE	(T_{ACTIVE} * B)
HFP	(T_{HFP} * B) – 8

B = 2, 2.25 or 3, depending on pixel format.

N = Number of lanes used

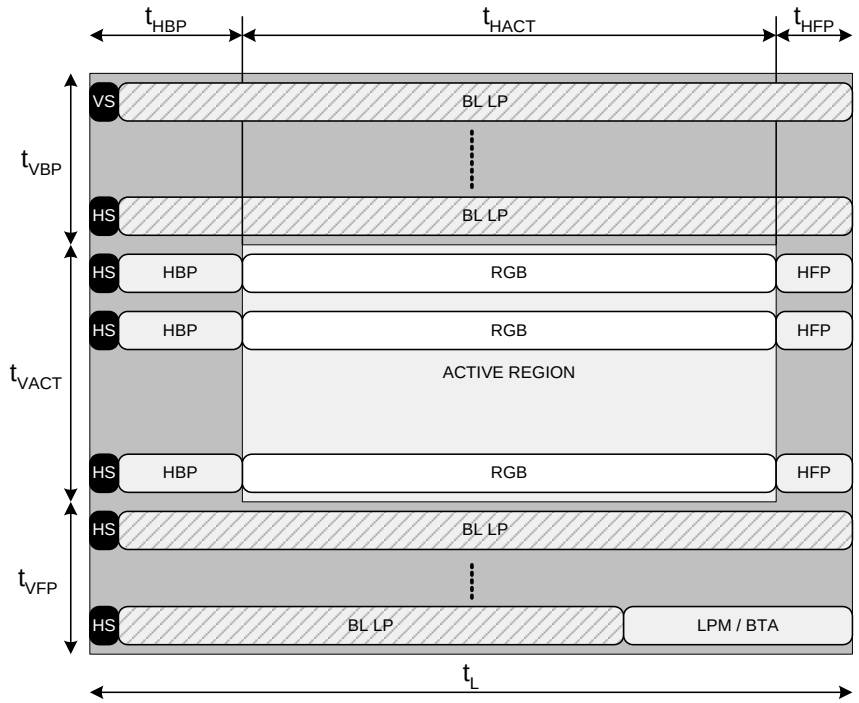
Table 115 Line Type Packet Sequences - Non-Burst with Sync Ends

LT	ID	Len	ID	Len	ID	Len	ID	Len	ID	Len	ID	Len
0	VS	0	EOT	0								
1	VE	0	EOT	0								
2	HS	0	EOT	0								
3	HS	0	BLNK	1	HE	0	BLNK	2	RGB	3	BLNK	4
4	HS	0	EOT	0								
5	HS	0	BLNK	1	HE	0	BLNK	2	RGB	3	BLNK	4

30.6.5 Non-Burst Mode (without ends)

This mode relaxes the requirement to mimic the generation of sync pulses and instead merely mandates that the start of the line is indicated and that the pixels appear at the same rate and in the same area of the raster as a tradition raster structure.

Figure 98 Timing Diagram for Video Non-Burst Mode



During the Vertical Active period, all packets should be concatenated into a single HS transmission for the duration of the Vertical Active period. As of writing, there is some debate as to whether the HFP blank packet is required, or even desirable. This may get replaced by a period of Blanking / Low Power mode.

The detail of how a single line is constructed from multiple packets is shown in the figure below. In addition, it will be necessary for the packet constructor on the display clock side of the display / DSI boundary FIFO to calculate the packet size for three different packets on the line. It is also suggested that there is a running-disparity adjustment made to the packet sizes so that the average pixel rate on the DSI side matches the display output rate.

Figure 99 Non-Burst Mode Packet Timing Detail

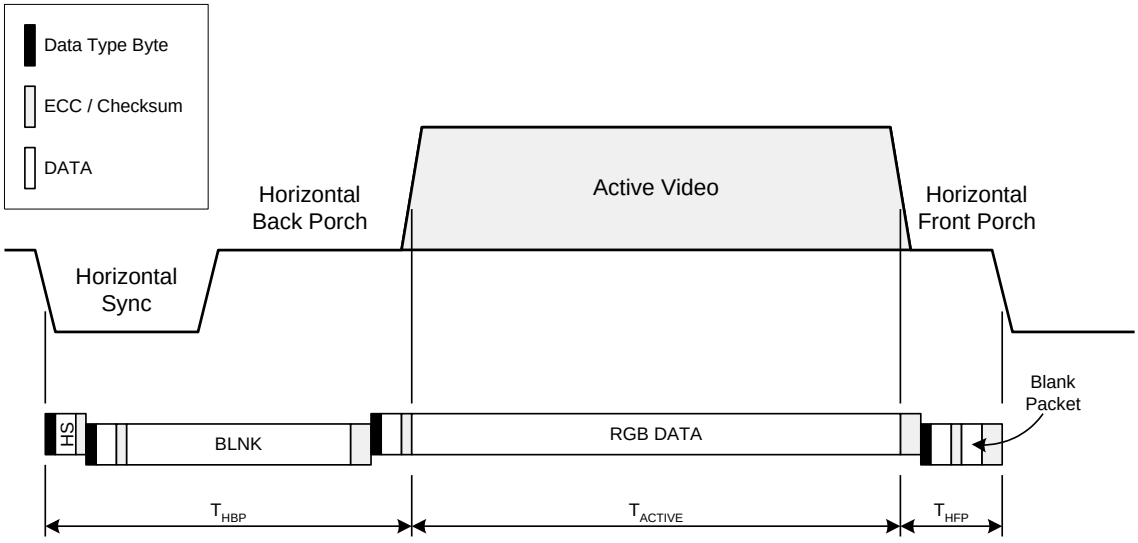


Table 116 Payload Size Table - Non-Burst Mode

Packet	Payload Size (Bytes)
HBP	(T_{HBP} * B) – 14
ACTIVE	(T_{ACTIVE} * B)
HFP	(T_{HFP} * B) – 8

B = 2, 2.25 or 3, depending on pixel format.

N = Number of Lanes used.

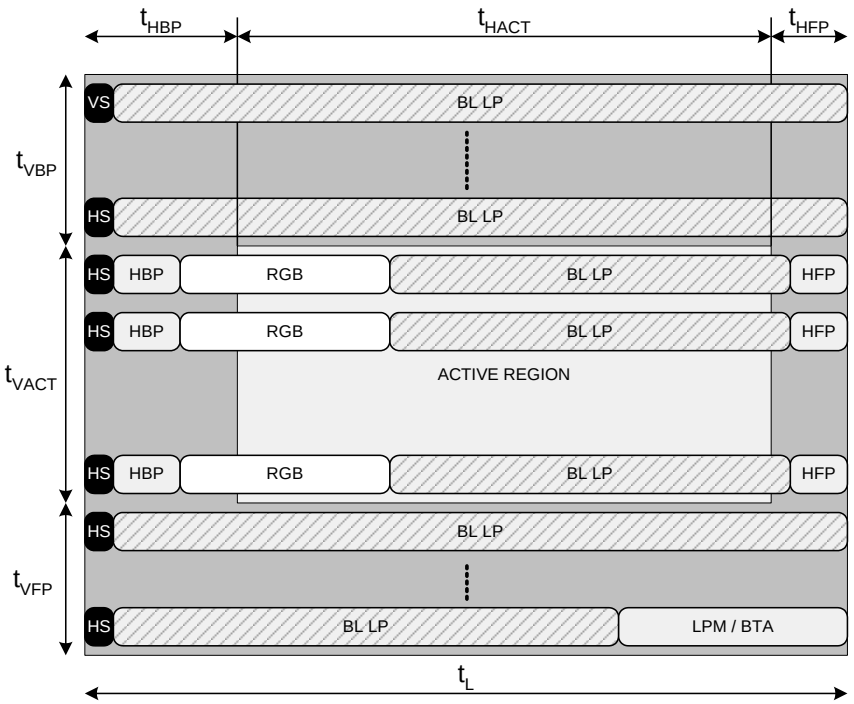
Table 117 Line Type Packet Sequences – Non Burst

LT	ID	Len	ID	Len	ID	Len	ID	Len	ID	Len	ID	Len
0	VS	0	EOT	0								
1	HS	0	EOT	0								
2	HS	0	EOT	0								
3	HS	0	BLNK	2	RGB	3	BLNK	4				
4	HS	0	EOT	0								
5	HS	0	BLNK	2	RGB	3	BLNK	4				

30.6.6 Burst Mode

In Burst Mode, the only attempt to match the raster structure is with the timing of the sync start events. The actual RGB pixel data is transmitted at whatever rate is convenient. This means that the HS, HBP and RGB packets become compressed with respect to the timing of the underlying raster. This allows some period of idle time each line that can be used for the transmission of other packets.

Figure 100 Timing Diagram for Video Burst Mode



During the Vertical Active period, packets on an individual line should be concatenated into a single HS transmission. However, unlike Non-Burst Mode, the bus should go idle – if possible – at the end of this transmission. This will allow additional non-video packets to be sent essentially simultaneously with the video stream. The NULL and HFP packets may be optional. Check the latest MIPI DSI specification for clarification.

Table 118 Line Type Packet Sequences - Bust Mode

LT	ID	Len	ID	Len	ID	Len	ID	Len	ID	Len	ID	Len
0	NULL	4	VS	0	EOT	0						
1	NULL	4	HS	0	EOT	0						
2	NULL	4	HS	0	EOT	0						
3	BLNK	4	HS	0	BLNK	2	RGB	3	EOT	0		
4	NULL	4	HS	0	EOT	0						
5	BLNK	4	HS	0	BLNK	2	RGB	3	EOT	0		

30.6.7 Command Mode Programming

There are two sources of command mode packet sequences:

- Display Controller
- Host Interface

Only one of these sources will be active at any particular time.

30.6.7.1 Command Mode from Host

In this mode of operation, all packets sent over the DSI interface are determined by Software. There may be hardware assistance in the generation of Error Correction Codes (ECC) and Cyclic Redundancy Checks (CRC), but all other data is passed un-altered to the DSI physical layer.

Host Packet Writes

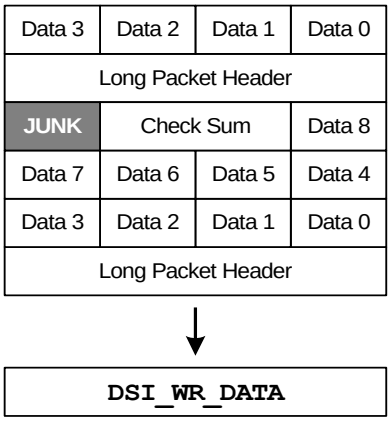
Packets are written to the HW by writing to the `DSI_WR_DATA` register. The data written is passed into the DSI Host transmit FIFO. If the data is a packet header and the `ECC_ENABLE` field in the `HOST_DSI_CONTROL` register is set to `ENABLE`, then the MSBs of the 32 bit packet header word is replaced with a Hardware computed ECC byte prior to being written into the FIFO. If this field is set to `DISABLE`, then no action is taken by the HW and the packet header is written unchanged.

If the packet is a long packet, then the packet payload should be written to the register after the packet header is written. If the `ECC_ENABLE` field of the `HOST_DSI_CONTROL` register is set to `ENABLE`, then the hardware computes the check-sum and appends it to the packet information. If this field is set to `DISABLE` then software must append the correctly computed check-sum to the packet data.

Rules:

1. Software should make sure packets are transmitted in their entirety and that once transmission starts, the FIFO contains enough data to finish a transition on a packet boundary. This can be achieved by only initiating a transmission – either explicitly, or indirectly – once all data required for a transmission has been written to the transmit FIFO.
2. Packets should be written such that the packet header is always written in one 32-bit word and is never split across writes. In other words, if the end of a packet does not fall on a 32-bit word boundary – if the payload has an odd length, for example – then the header for the next packet should not about the last packet, but should re-align with the register. See Figure 101 for an example.

Figure 101 Packet alignment for writes to DSI_WR_DATA



Host Packet Transmission

Control of when the Host write FIFO starts to drain (flush) can be programmed to come from several sources:

- Start Of Line signal from the Display Controller
- Once the FIFO has some programmable quantity of data in it.
- Explicit FLUSH bit in a control register

Start of Line (HOST_TX_TRIG_SRC == SOL **)**

This mode would typically be used when the Host is trying to emulate Video Mode to drive a raster-based display. Currently this mode of operation is a P1 feature.

Threshold (HOST_TX_TRIG_SRC == FIFO_LEVEL**)**

The FIFO can be programmed to automatically start transmission once a certain amount of data has been written to the FIFO. This is useful if the amount of data to be transmitted is already known and is essentially constant, or at least consistently more than the threshold amount. Software can then simply write to the `DSI_WR_DATA` register and transmission will start automatically. However, it should be ensured that if the threshold is set such that not all the data for a packet will have been written when the threshold is met, that software will write all the required data before the transmission has drained the FIFO completely.

Immediate (HOST_TX_TRIG_SRC == IMMEDIATE**)**

As the name implies – if a '1' is written to the `DSI_HOST_TRIGGER` field of the `DSI_TRIGGER` register, then transmission of the data in the Host write FIFO will start immediately. It it recommended that all data required for transmission is written to the `DSI_WR_DATA` register prior to setting this bit.

30.6.7.2 Command Mode from Display Controller

This mode of operation is intended to allow the display controller to write pixel data packets to a DBI-like device that does not require data to be sent in an isochronous raster structure like a DPI device. Of course, the data ***will*** be sent in an isochronous manner since it will be coming from the display controller, but the commands sent will be DCS control commands, rather than Video Mode timing and blanking packets.

Setup

There are four steps to operating in this mode:

1. Program the Peripheral display device using DCS commands via the Host Command interface. It is especially important to send the set_column_address, set_page_address and set_pixel_format commands. These effectively define the area to which we will be writing pixels.
2. Program the DSI_INIT_SEQ_CONTROL and DSI_INIT_SEQ_DATA_* registers in the DSI interface with appropriate values. Any commands required to be sent once per frame and *every* frame by way of set-up prior to the pixel data being sent should be put in here.
3. Program the DSI_DATA_FORMAT field of the DSI_CONTROL register to the required pixel format. Note that BIT18NP should not be programmed – see below on pixel format restrictions.
4. Program the DSI_PKT_SEQ_* registers. Program the registers in the flowing way:
   dd. 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.
   ee. Packet Sequence registers for Line Type 3 and 5 should be programmed with a DCS Long Write packet.
5. Program the DCS command ID register to have a write_start command associated with Line Type 5 (First line of active) and a write_continue command associated with Line Type 3 (all other active lines).
6. Enable the display. When Vertical syncs arrive from display, the initialization sequence should be sent and for every active line, the pixel data will be sent in a DCS Long Write packet.

Pixel format restrictions

The MIPI DCS specification allows for up to 6 different pixel data formats to be transmitted in a write_start or write_continue command. These pixel formats are listed in Table 119. Unfortunately, due to limitations in the current design of the DSI hardware, only 3 of the 6 formats can be supported.

The formats supported are also shown in the “supported” column of Table 119. Do not set BIT18NP as a pixel format. There is no DCS equivalent of this format, so undefined behavior may result if this format is set.

Table 119 DCS Pixel Format Support

DCS ID	DCS format	Supported	Name
0	reserved	N/A	-
1	3bpp	N	-
2	8bpp	N	-
3	12bpp	N	-
4	reserved	N/A	-
5	16bpp	Y	BIT16P
6	18bpp	Y	BIT18P
7	24bpp	Y	BIT24P

Simultaneous Host Command packets

It is necessary, from time to time, to send a DCS command to the display peripheral in a “side-band” fashion while the display controller is continuing to send DCS write commands with pixel information. To this end, Line Type 4 (First blank line) is reserved to indicate to the HW when it should attempt to send any DCS command packets requested by SW.

If it is desired to send a DCS command in this way, the DCS command packet should be written in its entirety (header, ECC, DCS command, payload, CSC) into the Host Command FIFO. The HOST_TX_TRIG_SRC field should then be set to IMMEDIATE.

The DSI HW will then send the entire contents of the FIFO out on the DSI interface on the first line of blanking. Sending the data on this line guarantees the Host packet transmission will have enough time to complete before the next packet generated by the Display Controller is generated.

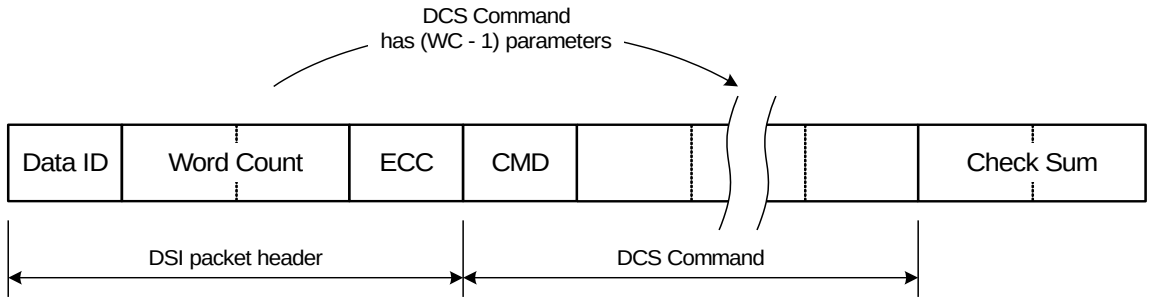
Note: There will be no BTA permitted in this mode, so no ACK, ACK with error report or read return data will be generated.

30.6.8 Display Command Set (DCS) Packets

DCS is a legacy control command set that is used to program various control registers present in typical LCD panels used in cell phones. Historically, the commands would be sent to the display over a MIPI DBI interface. Since MIPI DSI is meant to replace both DPI and DBI, it must also transport these control commands.

There are several DCS-specific commands in the MIPI DSI specification that can be used to send DCS commands to the Peripheral Display Device. However, the most useful is probably the DCS Long Write packet since it can be used to send commands with more than one parameter. The DCS command is embedded with an MIPI DSI Long Packet as follows: The DCS Command Byte and all the DCS Command Parameters (payload) are concatenated and sent in the Payload of the MIPI DSI Long Packet. The Word Count of the DSI packet conveys the total size of the DSC packet. As usual, a 2-byte CS footer is appended to the DSI Long Packet. A diagram of how a DCS Long Write packet is constructed is shown below in the next figure.

Figure 102 DCS command Placement in DSI Long Packet



30.6.9 Function Programming

30.6.9.1 ECC Generation

For precise details on the calculation of the ECC field of the packet header, reference should be made to the MIPI Alliance Standard for Display Serial Interface [2]. However, this is an overview of how the ECC can be created quite simply.

Each bit of the ECC byte is the result of XORing a number of bits from the packet header together. Which header bits contribute to which ECC bit is contained in a special table which can be used to calculate the ECC as follows:

```
const UCHAR ecc_parity[24] = { 0x07, 0x0b, 0x0d, 0x0e, 0x13, 0x15, 0x16, 0x19,
                               0x1a, 0x1c, 0x23, 0x25, 0x26, 0x29, 0x2a, 0x2c,
                               0x31, 0x32, 0x34, 0x38, 0x1f, 0x2f, 0x37, 0x3b
                             };
ULONG packet_header;
UCHAR ecc_byte;
UINT  i;

// Assume bottom 24 bits of packet_header
// contains header ID and byte count, then ...

ecc_byte = 0;
for (i = 0; i < 24; i++) {
  ecc_byte ^= ((packet_header >> i) & 1) ? ecc_parity[i] : 0x00;
```

```
}
packet header |= (ULONG)(ecc byte) << 24;
```

Note that the table in the DSI specification actually contains 64 entries. Since short packets have been fixed in length to be the same as a long packet header since the original specification was written, there will now always be just 24 data bits in the packet header, so only 24 entries are needed.

30.6.9.2 CRC Insertion

The DSI check sum used for long packets is derived using the generator polynomial $x^{16}+x^{12}+x^5+x^0$. Details of this can be found in section 9.6 of the MIPI DSI specification. To understand how the CRC is generated, it is easier to think of the packet data as consisting of a continuous serialized stream of bits, rather than a sequence of 8-bit bytes. When thought of in this way, it is relatively straight forward to generate the CRC. Reproduced below is an abridged version of the example C code contained in Appendix B of the MIPI DSI specification.

```
// Polynomial, bit reversed form (since DSI transmits LSB first) …
const unsigned short CRC16GenerationCode = 0x8408;

unsigned short CalculateCRC16( unsigned char *DataStream ptr, unsigned short NumberOfDataBytes)

{
  unsigned short ByteCounter;
  unsigned char  BitCounter;
  unsigned char  CurrentData;
  unsigned short CRC16Result = 0xFFFF;

  if (NumberOfDataBytes > 0) {
    for (ByteCounter = 0; ByteCounter < NumberOfDataBytes; ByteCounteer++) {
      CurrentData = DataStream_ptr[ByteCounter];
      for (BitCounter = 0; BitCounter < 8; BitCounter++) {
        if ((CRC16Result & 0x0001) ^ (CurrentData ^ 0x0001))
          CRC16Result = ((CRC16Result >> 1) & 0x7FFF) ^ CRCGenerationCode;
        else
          CRC16Result =  (CRC16Result >> 1) & 0x7FFF;
        CurrentData = (CurrentData >> 1) & 0x7F;
      }
    }
  }
  return CRC16Result;
}
```

This code may not be very high-performance due to the inner for loop which iterates over bits, rather than bytes. While it is instructive and could form the basis of a reference piece of code, it is not recommended where performance is important. There are many documented methods of performing this calculation in parallel in order to speed up the computations. These methods are beyond the scope of this document.

30.6.10 Read Data Return

DSI is a bi-directional interface. Data is returned from the peripheral display device only after the display controller has requested information by issuing a Bust Turn Around (BTA). All returned data is written into a FIFO that can be read using Host reads of a DSI register.

30.6.10.1 Reading Peripheral Registers

A typical application of BTA would be in the reading of a register from the display peripheral. This is achieved in the following way:

1. Set up the DSI interface to be in Host-driven command mode.
2. Set the DSI_MAX_THRESHOLD to 3.
3. Set the HOST_TX_TRIG_SRC field of the HOST_DSI_CONTROL register to FIFO_LEVEL.
4. Set the PKT_BTA field of the HOST_DSI_CONTROL register to ENABLE.
5. Write a DCS READ command packet (see section 8.8.8.2 of the MIPI DSI specification) into the Command FIFO by writing to the DSI_WR_DATA register.

This will result in the transmission of a DCS READ packet to the peripheral, the initiation of a BTA and – assuming the peripheral received the packet without error – the return of the requested data to the Host Read Return FIFO. The data can then be read from the FIFO by reading the DSI_RD_DATA register.

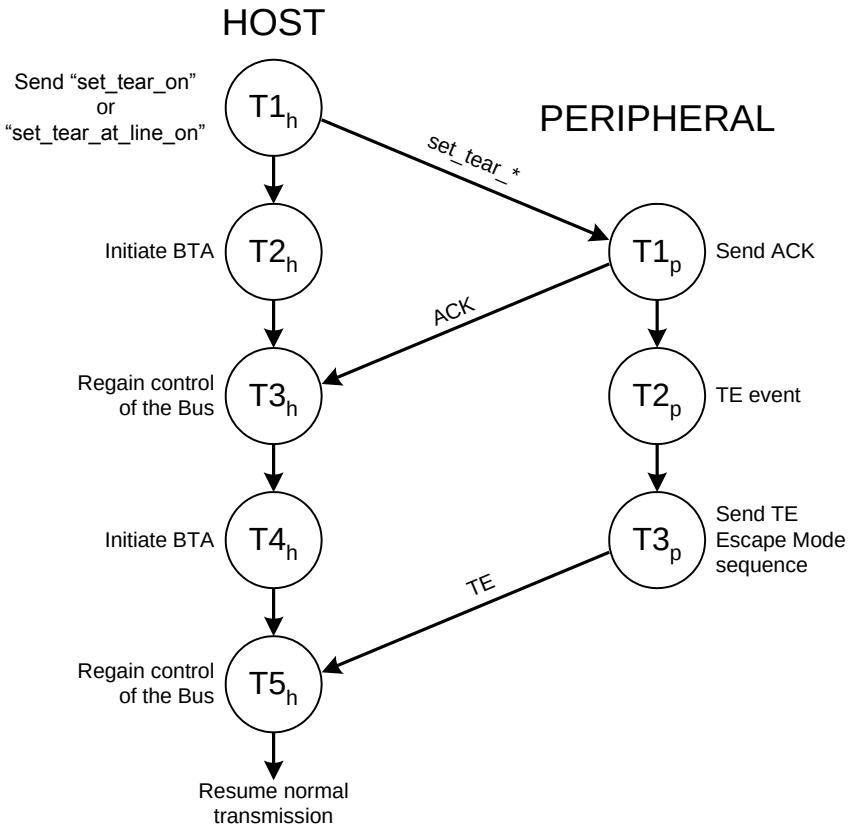
30.6.10.2 Bus Turn Around

- BTA is only supported for Host driven Command Mode interface. There will be no BTA during video mode transmission.
- Whether or not a BTA is initiated is controlled by the PKT_BTA field of the HOST_DSI_CONTROL register.
- The DSI Read Return FIFO is 4 bytes wide and 8 entries deep, so 32 bytes in total. this is enough to hold 8 short packets, or a mixture of short and long packets. Obviously, the length of long packets must be severely restricted.
- There will be no ECC or CS check performed by hardware on the read return data. Entire packets will simply be made available to software to perform whatever checks they desire.
- There will be the ability to increment a sync point counter on the arrival of the returned data or on the receipt of an error report in the event there was a problem with the read packet transaction.

30.6.11 Tearing Effect

In order to synchronize the update of a Command Mode display with data from the Host, a signal from the display can be sent to indicate when it is safe to proceed with the transmission of new data. This is the Tearing Effect reporting signal. Refer to section “8.12 TE Signaling in DSI” of the DSI specification for more details.

Figure 103. Tearing Effect State Transition Diagram



Programmatically, this is achieved in the following way:

1. Send SET_TEAR_ON or SET_TEAR_AT_LINE_ON command with the PKT_BTA field in the HOST_DSI_CONTROL register set to ENABLE. This is state $T1_h$.
2. Wait for ACK to come back from the peripheral. This is states $T2_h$ and $T3_h$.
3. Set the IMM_BTA field in the HOST_DSI_CONTROL register set to ENABLE. This will cause the D-PHY to go into BTA without having to send a command first. This is state $T4_h$.
4. Wait for TE return byte from the peripheral. This is state $T5_h$.

30.6.12 Error Reporting

There is no actual error recovery circuitry in the hardware, only error reporting. Any error that is reported via a protocol-level packet will be processed like all other return data and will be written to the data return FIFO for processing by the Host / Software.

Low level hardware errors such as bus contention will not be flagged, but will increment counters that can be queried by software so as to determine the reliability of the link.

30.6.12.1 Acknowledge with Error Report

The peripheral device (display) can be instructed to return an error report along with an acknowledge at the end of a transmission sequence. This is achieved by setting the PKT_BTA field in the HOST_DSI_CONTROL register to ENABLE. When this is done, there are several outcomes as shown in Table 120.

Table 120 Peripheral response to various conditions

Condition	Non-Read Packet	Read Packet
No error	ACK	Read Data
Corrected single bit error	ACK w/ERR	Read Data + ACK w/ERR
Non-corrected multi-bit error	ACK w/ERR	ACK w/ERR
SOT, EOT, VI ID or other D-PHY error	ACK w/ERR	ACK w/ERR

It is the responsibility of software to read this data from the packet returned in the packet return FIFO and process as required. No action will be taken by the hardware based on the error report returned in the ACK packet.

ACK Return

If a BTA request is enabled and there is no error, then the peripheral will return an ACK trigger to the DSI hardware. This is a single byte trigger and has the value 0x84. Since the return FIFO is 32 bits wide, the 0x84 value will be placed in the bottom 8 bits of the FIFO.

Read Data

If a read command packet is sent, then BTA request should be enabled to allow the peripheral to return the read data. If no error occurs in the transmission of the read packet, or a corrected single bit error occurs, then the read data will be returned to the host DSI hardware in the form of a read packet. The precise form will depend on the type of read packet transmitted. Refer to the MIPI DSI specification, section 8.10 for details.

If a corrected error occurs during the transmission of the read command, the requested read data will be returned in the normal way, but an ACK with Error report packet will be appended to the read return data packet.

ACK with Error Report

The error report is contained in the 16 payload bits of a special short packet returned by the peripheral. The bits allocations of the packet data are detailed in section 8.9.5 of the MIPI DSI specification but are repeated here for convenience.

Table 121 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

30.6.13 Time Outs

There are three compulsory and one optional time out counters required by the MIPI DSI specification for Processors (display controller). Each timer will consist of a simple counter which will count DSI byte clocks. The counters will reset to 0 on an event and will then simply increment their count every DSI byte clock cycle. If the event that the time out is protecting occurs before the time out reaches its terminal count, the counter will simply stop counting and hold its value. If the counter reaches software programmed maximum count before the expected event occurs, the counter will stop counting and hold its count value. Any action that is required by the MIPI DSI specification upon reaching the time out will also be performed by the hardware.

Table 122 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 122, but will also cause a tally counter to increment so that software can read the register and determine if any of the time outs have occurred. Software will be able to reset these tally counters to 0 by writing to the tally register. The value written will be irrelevant. The PR_TO will report its operation in a different manner. When the PR_TO counter is actually counting, there will be a status bit that reads as ‘0’. When the PR_TO terminal count is reached and time out ends, the status bit will become ‘1’. This will allow SW to determine when the peripheral has been reset.

Table 123 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

30.7 MIPI-DSI Registers

30.7.1 DSI_INCR_SYNCPT_0

Offset: 000h | Read/Write: R/W | Reset: 0b0000000000000000

Bit	Reset	Description
15:8	0x0	COND: Condition mapped from raise/wait 0 = IMMEDIATE 1 = OP_DONE 2 = RD_DONE 3 = REG_WR_SAFE 4 = COND_4 5 = COND_5 6 = COND_6 7 = COND_7 8 = COND_8 9 = COND_9 10 = COND_10 11 = COND_11 12 = COND_12 13 = COND_13 14 = COND_14 15 = COND_15
7:0	0x0	INDX: syncpt index value

30.7.2 DSI_INCR_SYNCPT_CNTRL_0

Offset: 001h | Read/Write: R/W | Reset: 0b0xxxxxxx0

Bit	Reset	Description
8	0x0	INCR_SYNCPT_NO_STALL: If NO_STALL is 1, then when fifos are full, INCR_SYNCPT methods will be dropped and the INCR_SYNCPT_ERROR[COND] bit will be set. If NO_STALL is 0, then when fifos are full, the client host interface will be stalled.
0	0x0	INCR_SYNCPT_SOFT_RESET: If SOFT_RESET is set, then all internal state of the client syncpt block will be reset. To do soft reset, first set SOFT_RESET of all host1x clients affected, then clear all SOFT_RESETs.

30.7.3 DSI_INCR_SYNCPT_ERROR_0

Offset: 002h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	COND_STATUS: COND_STATUS[COND] is set if the fifo for COND overflows. This bit is sticky and will remain set until cleared. Cleared by writing 1.

30.7.4 DSI_CTXSW_0

Context switch register (common to all modules). Includes the current channel/class (which is writable by SW) and the next channel/class (which the hardware sets when it receives a context switch).

Context switch works like this:

Any context switch request triggers an interrupt to the host and causes the new channel/class to be stored in NEXT_CHANNEL/NEXT_CLASS (see vmod/chexample). SW sees that there is a context switch interrupt and does the necessary operations to make the module ready to receive traffic from the new context. It clears the context switch interrupt

and writes CURR_CHANNEL/CLASS to the same value as NEXT_CHANNEL/CLASS, which causes a context switch acknowledge packet to be sent to the host. This completes the context switch and allows the host to continue sending data to the module. Context switches can also be pre-loaded. If CURR_CLASS/CHANNEL are written and updated to the next CLASS/CHANNEL before the context switch request occurs, an acknowledge will be generated by the module and no interrupt will be triggered. This is one way for software to avoid dealing with context switch interrupts.

Another way to avoid context switch interrupts is to set the AUTO_ACK bit. This bit tells the module to automatically acknowledge any incoming context switch requests without triggering an interrupt. CURR_* and NEXT_* will be updated by the module so they will always be current.

Offset: 008h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx11111x0000000000

Bit	R/W	Reset	Description
31:28	RO	X	NEXT_CHANNEL: Next requested channel
25:16	RO	X	NEXT_CLASS: Next requested class
15:12	RW	0xf	CURR_CHANNEL: Current working channel, reset to 'invalid'
11	RW	0x1	AUTO_ACK: Automatically acknowledge any incoming context switch requests 0 = MANUAL 1 = AUTOACK
9:0	RW	0x0	CURR_CLASS: Current working class

Note: **PACKET DEFINITIONS**

Packet for Display Controller -> DSI communication:

Line Type is used to convey the type of video line from the display controller to the DSI block. The line type implies the generation of one of the associated packet sequences as defined in the DSI packet sequence registers (see below). Used to construct packet headers. All packet headers are now 32-bits wide regardless of whether they are short or long packets. Used for validation infrastructure only.

30.7.5 DSI_DSI_RD_DATA_0

DSI read return data.

Offset: 009h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	DSI_RD_DATA: Each read to this register will pop 32 bits from the 32 bit wide read return data FIFO. FIFO has NV_DSI_HOST_DATA_RETURN_FIFO_DEPTH entries.

30.7.6 DSI_DSI_WR_DATA_0

Host FIFO write input.

Offset: 00ah | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	DSI_WR_DATA: Each write to this register will puch 32 bits into the 32 bit wide Host data FIFO. FIFO has NV_DSI_HOST_DATA_FIFO_DEPTH entries

30.7.7 DSI_DSI_POWER_CONTROL_0

Display Power Control. DSI Enable.

Offset: 00bh | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
0	0x0	LEG_DSI_ENABLE: DSI interface Enable 0 = DISABLE : Disable DSI 1 = ENABLE : Enable DSI

30.7.8 DSI_INT_ENABLE_0

Interrupt enable register.

Offset: 00ch | Read/Write: R/W | Reset: 0b1

Bit	Reset	Description
0	0x1	CTXSW_INT_ENABLE: Context Switch Interrupt Enable 0 = DISABLE : interrupt disabled 1 = ENABLE : interrupt enabled

30.7.9 DSI_INT_STATUS_0

Interrupt Status register.

Offset: 00dh | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
0	0x0	CTXSW_INT: Context switch interrupt status (clear on write)

30.7.10 DSI_INT_MASK_0

Interrupt Mask Setting bits in this register masked the corresponding interrupt but does not clear a pending interrupt and does not prevent a pending interrupt to be generated. Masking an interrupt also does not clear a pending interrupt status and does not prevent a pending interrupt status to be generated.

Offset: 00eh | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
0	0x0	CTXSW_INT_MASK: Context Switch Interrupt Mask 0 = MASKED : interrupt masked 1 = NOTMASKED : interrupt not masked

30.7.11 DSI_HOST_DSI_CONTROL_0

DSI control register when input is from HOST

Offset: 00fh | Read/Write: R/W | Reset: 0b00x000xx00xx0000000011

Bit	Reset	Description
21	0x0	FIFO_STAT_RESET: write only bit to clear FIFO underflow/overflowflags. If there new underflow/overflow event occurs during thesame time, current access cant clear the status bits
20	0x0	CRC_RESET: Write only bit. When written with a 1, causes the Verification CRC generator to reset to 0xFFFF_FFFF. If written with 0, it has no effect.

Bit	Reset	Description
18:16	0x0	DSI_PHY_CLK_DIV: Phy clock divider value for byte clock 0 = DIV1 1 = DIV2
13:12	0x0	HOST_TX_TRIG_SRC: Controls the source of the trigger to start sending packets. 0 = SOL : Start of Line signal from the Display Controller. 1 = FIFO_LEVEL : How full the FIFO is. Level determined elsewhere. 2 = IMMEDIATE : Determined by a write to the DSI_HOST_TRIGGER field of the DSI_TRIGGER register
9:8	0x0	DSI_ULTRA_LOW_POWER: Ultra Low Power 0 = NORMAL 1 = ENTER_ULPM 2 = EXIT_ULPM
7	0x0	PERIPH_RESET: Initiate an Escape Mode Peripheral Reset. 0 = DISABLE : Causes an Escape Mode Command to be sent to reset the 1 = ENABLE : external display device. Also starts the PR_TO counter. PR_TO state can be checked with the P_RESET_STATUS field in the DSI_TO_TALLY register. HW clears this bit upon completion of issuing "Trigger Reset" OR called "Reset Entry" command is sent out.
6	0x0	RAW_DATA: Host raw data mode. In this mode. All data is sent exactly as written. No attempt to decode packet headers is made. This bit will aldo override the function of the ECC and CS ENABLE fields. No ECC or CS will be generated. This mode is intended as a debug / CYA 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 0 = HOST : Write data to the small host data fifo only. 1 = VIDEO : Write data to both the host and video line store fifo, in series.
3	0x0	IMM_BTA: Generate BTA immediately, eg. for Tearing Effect reporting. Note: This will generate a BTA and pass control of the D-PHY to the remote peripheral without the need to send any packet. Once the remote peripheral has responded and relinquished control of the bus, this bit will be cleared by the HW. 0 = DISABLE : Do not generate BTA 1 = ENABLE : Generate BTA immediately, without waiting for a packet.
2	0x0	PKT_BTA: Generate BTA at the end of Host packets 0 = DISABLE : Do not generate BTA 1 = ENABLE : Generate BTA after the next packet is sent.
1	0x1	CS_ENABLE: enable Hardware Check Sum (CS) for Host packets Note: when CS is disabled, Host is responsible for generating proper CRC and adding the 2-byte CRC to the end of the packet after the payload. 0 = DISABLE : disable HW generation of CS (Host must calculate CS). 1 = ENABLE : enable HW generation of CS.
0	0x1	ECC_ENABLE: enable Hardware Error Correction Code (ECC) for Host packets Note: when ECC is disabled, Host is responsible for generating proper ECC byte for header 0 = DISABLE : disable HW generation of ECC (Host must calculate ECC). 1 = ENABLE : enable HW generation of ECC.

30.7.12 DSI_DSI_CONTROL_0

General DSI Control register

Offset: 010h | Read/Write: R/W | Reset: 0b0xxxxxxxxxx0xx00xx00xx00xx000000

Bit	Reset	Description
31	0x0	DSI_DBG_ENABLE: Control signal to turn off clock monitoring when enabled for debug, on every DSI byte clock debug signal toggles. Also TX CRC computation aid will turn on
20	0x0	DSI_HS_CLK_CTRL: Control for the HS clock lane 0 = CONTINUOUS : HS clock is on all the time. 1 = TX_ONLY : HS clock is only active during HS transmissions.
17:16	0x0	DSI_VIRTUAL_CHANNEL: Virtual channel ID Virtual channel is sent as part of packet header and used to distinguish multiple displays.
13:12	0x0	DSI_DATA_FORMAT: Pixel Data format transmitted. Note that although the pixel format is specified in the packet header ID for RGB data packets, this information is ignored by the HW. Only the information used in this register is used in the construction of RGB data packets. 0 = BIT16P : 16 bpp RGB Packed. 2 bytes used per pixel 1 = BIT18NP : 18 bpp RGB Not-packed. 3 bytes used per pixel 2 = BIT18P : 18 bpp RGB Packed. 2.25 bytes used per pixel 3 = BIT24P : 24 bpp RGB Packed. 3 bytes used per pixel
9:8	0x0	VID_TX_TRIG_SRC: Controls the source of the trigger to start sending packets. 0 = SOL : Start of Line signal from the Display Controller. 1 = FIFO_LEVEL : How full the FIFO is. Level determined elsewhere. 2 = IMMEDIATE : Determined by a write to the DSI_VID_TRIGGER field of the DSI_TRIGGER register
5:4	0x0	DSI_NUM_DATA_LANES: Number of D-PHY data lanes used by Display for HS transmission. 0 = ONE : 1 data lane. 1 = TWO : 2 data lanes. 2 = THREE : 3 data lanes. (NOTE: Currently, this is illegal) 3 = FOUR : 4 data lanes. (NOTE: Currently, this is illegal)
3	0x0	VID_DCS_ENABLE: Enable for insertion of DCS commands during Display Controller generated packets. When enabled, the DCS commands defined in the LT3_DCS_CMD and LT5_DCS_CMD fields of the DSI_DCS_CMDS register will be inserted in long packets defined in packet sequence 3 and 5. 0 = DISABLE : No DCS commands will be inserted. 1 = ENABLE : DCS command IDs will be inserted as described above.
2	0x0	DSI_VID_SOURCE: Source of video pixels 0 = DISPLAY_0 : pixels come from "display" 1 = DISPLAY_1 : pixels come from "displayb"
1	0x0	DSI_VID_ENABLE: Video DSI interface Enable 0 = DISABLE : disable 1 = ENABLE : enable
0	0x0	DSI_HOST_ENABLE: Host DSI interface Enable 0 = DISABLE : disable 1 = ENABLE : enable

30.7.13 DSI_DSI_SOL_DELAY_0

Number of byte-clock counts to wait after reception of.

Offset: 011h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	SOL_DELAY: Start Of Line before generating output packets.

30.7.14 DSI_DSI_MAX_THRESHOLD_0

Maximum threshold registers for DSI related packets.

Offset: 012h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	MAX_THRESHOLD: Start draining fifo once this threshold is met This register can be used for DBI mode when line packet data exceeds the size of the data fifo.

30.7.15 DSI_DSI_TRIGGER_0

Manual transmissions trigger register.

Offset: 013h | Read/Write: R/W | Reset: 0bxx

Bit	Reset	Description
1	X	DSI_HOST_TRIGGER: A 1 written to this bit will start host transmission when HOST_DSI_CONTROL.HOST_TX_TRIG_SRC is set to IMMEDIATE HW auto clears on operation completion
0	X	DSI_VID_TRIGGER: A 1 written to this bit will start video transmission. when DSI_CONTROL.VID_TX_TRIG_SRC is set to IMMEDIATE HW auto clears on operation completion

30.7.16 DSI_DSI_TX_CRC_0

Transmission CRC.

Offset: 014h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	TX_CRC: Long Packet CRC appended to the end of long packets. This CRC is that result of generating a CRC from all transmitted bytes. If DSI_HOST_ENABLE == ENABLE, then a CRC is generated from all bytes transmitted from the Host interface in Command Mode. If DSI_VID_ENABLE == ENABLE, then a CRC is generated from all bytes transmitted during a frame of video when in Video Mode. Note that the HW will capture the CRC into a separate internal register so that it can continue to calculate the CRC for the next frame without having to wait for SW to read the current result.

30.7.17 DSI_DSI_STATUS_0

DSI Status register.

Offset: 015h | Read/Write: RO | Reset: 0bxxxxxxxxxx

Bit	Reset	Description
9	X	LB_UNDERFLOW: Indicates that Line buffer underflow event happened
8	X	LB_OVERFLOW: Indicates that Line buffer overflow event happened
4:0	X	RD_FIFO_COUNT: Count of how many data words are left in the Host Read Data Return FIFO. Typically, SW knows how much data to read from the DSI_RD_DATA register after a Read packet / BTA has been sent / requested, since these transactions are initiated by SW. However, under error conditions, insufficient data may have been read from the FIFO. SW should therefore check this field to make sure it is 0 after reading all the information it expected to get.

30.7.18 DSI_DSI_INIT_SEQ_CONTROL_0

Initialisation Sequence Registers

DSI Initialization Sequence Control.

Offset: 01ah | Read/Write: R/W | Reset: 0bxxxxxxxxxxxx0

Bit	Reset	Description
12:8	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 16 Invalid programmable values: 17 to 31
0	0x0	DSI_SEND_INIT_SEQUENCE: Send Initialization Sequence (IS) 0 = DISABLE : 1 = ENABLE :

30.7.19 DSI_DSI_INIT_SEQ_DATA_0_0

DSI Init Sequence Write Data 0.

Offset: 01bh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_0: DSI Init Sequence Write Data bits 31-0

30.7.20 DSI_DSI_INIT_SEQ_DATA_1_0

DSI Init Sequence Write Data 1.

Offset: 01ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_1: DSI Init Sequence Write Data bits 31-0

30.7.21 DSI_DSI_INIT_SEQ_DATA_2_0

DSI Init Sequence Write Data 2.

Offset: 01dh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_2: DSI Init Sequence Write Data bits 31-0

30.7.22 DSI_DSI_INIT_SEQ_DATA_3_0

DSI Init Sequence Write Data 3.

Offset: 01eh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_3: DSI Init Sequence Write Data bits 31-0

30.7.23 DSI_DSI_PKT_SEQ_0_LO_0

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 second digit denotes the packet number within each sequence. For example, field PKT_23_ID, defines the ID for packet 3 in sequence 2.

DSI Packet Sequence 0 LO half

Offset: 023h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
30	X	SEQ_0_FORCE_LP: For packet sequence 0, forces the D-PHY to go to LP mode after the last packet of the sequence has been transmitted. 0 = DISABLE 1 = ENABLE
29	X	PKT_02_EN: Packet 2 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_02_ID: Packet 2 Packet ID
22:20	X	PKT_02_SIZE: Packet 2 size pointer
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

Bit	Reset	Description
2:0	X	PKT_00_SIZE: Packet 0 size pointer

30.7.24 DSI_DSI_PKT_SEQ_0_HI_0

DSI Packet Sequence 0 HI half.

> **Note:** Line Type 0 is associated with the first line in the frame and should contain a packet sequence that starts with a VS packet.

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

Bit	Reset	Description
29	X	PKT_05_EN: Packet 5 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_05_ID: Packet 5 Packet ID
22:20	X	PKT_05_SIZE: Packet 5 size pointer
19	X	PKT_04_EN: Packet 4 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_04_ID: Packet 4 Packet ID
12:10	X	PKT_04_SIZE: Packet 4 size pointer
9	X	PKT_03_EN: Packet 3 enable 0 = DISABLE 1 = ENABLE
8:3	X	PKT_03_ID: Packet 3 Packet ID
2:0	X	PKT_03_SIZE: Packet 3 size pointer

30.7.25 DSI_DSI_PKT_SEQ_1_LO_0

DSI Packet Sequence 1 LO half.

> **Note:** Line Type 1 is associated with the last line of Vertical Sync and should contain a packet sequence that starts with a VE packet.

Offset: 025h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
30	X	SEQ_1_FORCE_LP: For packet sequence 1, forces the D-PHY to go to LP mode after the last packet of the sequence has been transmitted. 0 = DISABLE 1 = ENABLE
29	X	PKT_12_EN: Packet 2 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_12_ID: Packet 2 Packet ID

Bit	Reset	Description
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

30.7.26 DSI_DSI_PKT_SEQ_1_HI_0

DSI Packet Sequence 1 HI half

Offset: 026h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
29	X	PKT_15_EN: Packet 5 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_15_ID: Packet 5 Packet ID
22:20	X	PKT_15_SIZE: Packet 5 size pointer
19	X	PKT_14_EN: Packet 4 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_14_ID: Packet 4 Packet ID
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

30.7.27 DSI_DSI_PKT_SEQ_2_LO_0

DSI Packet Sequence 2 LO half.

Note: Line Type 2 is associated with any vertical blank line except the first one after active and should contain a packet sequence that starts with an HS packet.

Offset: 027h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
30	X	SEQ_2_FORCE_LP: For packet sequence 2, forces the D-PHY to go to LP mode after the last packet of the sequence has been transmitted. 0 = DISABLE 1 = ENABLE
29	X	PKT_22_EN: Packet 2 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_22_ID: Packet 2 Packet ID
22:20	X	PKT_22_SIZE: Packet 2 size pointer
19	X	PKT_21_EN: Packet 1 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_21_ID: Packet 1 Packet ID
12:10	X	PKT_21_SIZE: Packet 1 size pointer
9	X	PKT_20_EN: Packet 0 enable 0 = DISABLE 1 = ENABLE
8:3	X	PKT_20_ID: Packet 0 Packet ID
2:0	X	PKT_20_SIZE: Packet 0 size pointer

30.7.28 DSI_DSI_PKT_SEQ_2_HI_0

DSI Packet Sequence 2 HI half.

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

Bit	Reset	Description
29	X	PKT_25_EN: Packet 5 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_25_ID: Packet 5 Packet ID
22:20	X	PKT_25_SIZE: Packet 5 size pointer
19	X	PKT_24_EN: Packet 4 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_24_ID: Packet 4 Packet ID
12:10	X	PKT_24_SIZE: Packet 4 size pointer
9	X	PKT_23_EN: Packet 3 enable 0 = DISABLE 1 = ENABLE
8:3	X	PKT_23_ID: Packet 3 Packet ID
2:0	X	PKT_23_SIZE: Packet 3 size pointer

30.7.29 DSI_DSI_PKT_SEQ_3_LO_0

DSI Packet Sequence 3 LO half.

> **Note:** Line Type 3 is associated with any active line except the first one and should contain a packet sequence that starts with an HS packet and includes an RGB data packet.

Offset: 029h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
30	X	SEQ_3_FORCE_LP: For packet sequence 3, forces the D-PHY to go to LP mode after the last packet of the sequence has been transmitted. 0 = DISABLE 1 = ENABLE
29	X	PKT_32_EN: Packet 2 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_32_ID: Packet 2 Packet ID
22:20	X	PKT_32_SIZE: Packet 2 size pointer
19	X	PKT_31_EN: Packet 1 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_31_ID: Packet 1 Packet ID
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

30.7.30 DSI_DSI_PKT_SEQ_3_HI_0

DSI Packet Sequence 3 HI half.

Offset: 02ah | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
29	X	PKT_35_EN: Packet 5 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_35_ID: Packet 5 Packet ID
22:20	X	PKT_35_SIZE: Packet 5 size pointer
19	X	PKT_34_EN: Packet 4 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_34_ID: Packet 4 Packet ID

Bit	Reset	Description
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

30.7.31 DSI_DSI_PKT_SEQ_4_LO_0

DSI Packet Sequence 4 LO half.

Note: Line Type 4 is associated with the first vertical blanking line after the last active line and should contain a packet sequence that starts with an HS packet. Ordinarily, this packet sequence should be identical to sequence 2.

Offset: 02bh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
30	X	SEQ_4_FORCE_LP: For packet sequence 4, forces the D-PHY to go to LP mode after the last packet of the sequence has been transmitted. 0 = DISABLE 1 = ENABLE
29	X	PKT_42_EN: Packet 2 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_42_ID: Packet 2 Packet ID
22:20	X	PKT_42_SIZE: Packet 2 size pointer
19	X	PKT_41_EN: Packet 1 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_41_ID: Packet 1 Packet ID
12:10	X	PKT_41_SIZE: Packet 1 size pointer
9	X	PKT_40_EN: Packet 0 enable 0 = DISABLE 1 = ENABLE
8:3	X	PKT_40_ID: Packet 0 Packet ID
2:0	X	PKT_40_SIZE: Packet 0 size pointer

30.7.32 DSI_DSI_PKT_SEQ_4_HI_0

DSI Packet Sequence 4 HI half.

Offset: 02ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
29	X	PKT_45_EN: Packet 5 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_45_ID: Packet 5 Packet ID
22:20	X	PKT_45_SIZE: Packet 5 size pointer
19	X	PKT_44_EN: Packet 4 enable 0 = DISABLE 1 = ENABLE
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

30.7.33 DSI_DSI_PKT_SEQ_5_LO_0

DSI Packet Sequence 4 LO half.

Note: Line Type 5 is associated with the first active line and should contain a packet sequence that starts with an HS packet and includes an RGB data packet. Ordinarily, this packet sequence should be identical to sequence 3.

Offset: 02dh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
30	X	SEQ_5_FORCE_LP: For packet sequence 4, forces the D-PHY to go to LP mode after the last packet of the sequence has been transmitted. 0 = DISABLE 1 = ENABLE
29	X	PKT_52_EN: Packet 2 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_52_ID: Packet 2 Packet ID
22:20	X	PKT_52_SIZE: Packet 2 size pointer
19	X	PKT_51_EN: Packet 1 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_51_ID: Packet 1 Packet ID

Bit	Reset	Description
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

30.7.34 DSI_DSI_PKT_SEQ_5_HI_0

DSI Packet Sequence 5 HI half.

Offset: 02eh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
29	X	PKT_55_EN: Packet 5 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_55_ID: Packet 5 Packet ID
22:20	X	PKT_55_SIZE: Packet 5 size pointer
19	X	PKT_54_EN: Packet 4 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_54_ID: Packet 4 Packet ID
12:10	X	PKT_54_SIZE: Packet 4 size pointer
9	X	PKT_53_EN: Packet 3 enable 0 = DISABLE 1 = ENABLE
8:3	X	PKT_53_ID: Packet 3 Packet ID
2:0	X	PKT_53_SIZE: Packet 3 size pointer

30.7.35 DSI_DSI_DCS_CMDS_0

Note: DCS COMMAND AND PACKET LENGTH REGISTERS

DCS command IDs used for Line Type 3 and 5. These command IDs are inserted at the start of the data payload of DCS long packets when the Display Controller is being used to transmit DCS commands.

Offset: 033h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:8	X	LT5_DCS_CMD: DCS command for Line Type 5.
7:0	X	LT3_DCS_CMD: DCS command for Line Type 3.

30.7.36 DSI_DSI_PKT_LEN_0_1_0

DSI packet lengths 0 and 1.

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

Bit	Reset	Description
31:16	X	LENGTH_1: Packet length 1 (in bytes)
15:0	X	LENGTH_0: Packet length 0 (in bytes)

30.7.37 DSI_DSI_PKT_LEN_2_3_0

DSI packet lengths 2 and 3.

Offset: 035h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	LENGTH_3: Packet length 3 (in bytes)
15:0	X	LENGTH_2: Packet length 2 (in bytes)

30.7.38 DSI_DSI_PKT_LEN_4_5_0

DSI packet lengths 4 and 5.

Offset: 036h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	LENGTH_5: Packet length 5 (in bytes)
15:0	X	LENGTH_4: Packet length 4 (in bytes)

30.7.39 DSI_DSI_PKT_LEN_6_7_0

DSI packet lengths 6 and 7.

Offset: 037h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	LENGTH_7: Packet length 7 (in bytes)
15:0	X	LENGTH_6: Packet length 6 (in bytes)

30.7.40 DSI_DSI_PHY_TIMING_0_0

Physical Interface Timing Registers

DSI D-PHY timing register 0.

Offset: 03ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:24	X	DSI_THSDEXIT: Time to drive LP11 after HS
23:16	X	DSI_THSTRAIL: Time to drive HS flipped bit at EOT
15:8	X	DSI_TDATZERO: Time to drive HS0 before SOT
7:0	X	DSI_THSPREPR: Time to drive LP00 before HS data

30.7.41 DSI_DSI_PHY_TIMING_1_0

DSI D-PHY timing register 1.

Offset: 03dh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:24	X	DSI_TCLKTRAIL: Time to drive HS0 before clock goes to LP1 1
23:16	X	DSI_TCLKPOST: Time to drive Clock after the last HS data
15:8	X	DSI_TCLKZERO: Time to drive LP00 before HS clock
7:0	X	DSI_TTLPX: LP period

30.7.42 DSI_DSI_PHY_TIMING_2_0

DSI D-PHY timing register 2.

Offset: 03eh | Read/Write: R/W | Reset: 0b00xxxxxxxx

Bit	Reset	Description
9:8	0x0	DSI_TCLKPRE: Time to run clock before turning on data lane
7:0	X	DSI_TWAKEUP: LP period when exiting ULPM. in units of 512 byte clocks.

30.7.43 DSI_DSI_BTA_TIMING_0

DSI D-PHY Bus-Turn-Around timing.

Offset: 03fh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	DSI_TTAGET: Time to Drive LP00 at end of BTA (5 * TTLPX)
15:8	X	DSI_TTASURE: Time to Receive LP00 at end of BTA (2 * TTLPX)
7:0	X	DSI_TTAGO: Time to drive LP00 at start of BTA (4 * TTLPX)

30.7.44 DSI_DSI_TIMEOUT_0_0

Note: CONTENTION RECOVERY TIMERS
These registers control the length of time - in units of 512 DSI byte clocks - that can elapse before the hardware will decide that a bus contention error has occurred. There are several counters to deal with various forms of error that may occur. For details, refer to MIPI DSI Spec. section 7.2.2

DSI Time out terminal count register 0.

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

Bit	Reset	Description
31:16	X	LRXH_TO: Low Power Receive (Host) Time Out terminal count
15:0	X	HTX_TO: High Speed Transmit Time Out terminal count

30.7.45 DSI_DSI_TIMEOUT_1_0

DSI Time out terminal count register 1.

Offset: 045h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	PR_TO: Peripheral Reset duration.
15:0	X	TA_TO: Turn Around Time Out terminal count

30.7.46 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: 046h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

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

30.7.47 DSI_PAD_CONTROL_0

Physical Pad Control Registers

DSI PHY configuration register SW can program this register to work with different panels.

Offset: 04bh | Read/Write: R/W | Reset: 0b1xxxxxxxxx111xxxxxxxxxxxxxxxx

Bit	Reset	Description
28	0x1	DSI_PAD_PULLDN_ENAB
27	X	DSI_PAD_REV_CLK
26:24	X	DSI_PAD_SLEWUPADJ
22:20	X	DSI_PAD_SLEWDNADJ
19	X	DSI_PAD_PREEMP_EN
18	0x1	DSI_PAD_PDIO_CLK
17:16	0x3	DSI_PAD_PDIO
15:14	X	DSI_PAD_LPUPADJ
13:12	X	DSI_PAD_LPDNADJ
10:8	X	DSI_PAD_OUTADJCLK
6:4	X	DSI_PAD_OUTADJ1
3	X	DSI_PAD_BANDWD_IN
2:0	X	DSI_PAD_OUTADJ0

30.7.48 DSI_PAD_CONTROL_CD_0

Contention detection logic enable signals.

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

Bit	Reset	Description
6:4	X	DSI_PAD_CDDNADJ
2	X	DSI_PAD_CD_EN_CLK
1:0	X	DSI_PAD_CD_EN

30.7.49 DSI_PAD_CD_STATUS_0

Contention detection status from MIPI PAD.

Offset: 04dh | Read/Write: RO | Reset: 0bxxxxxx

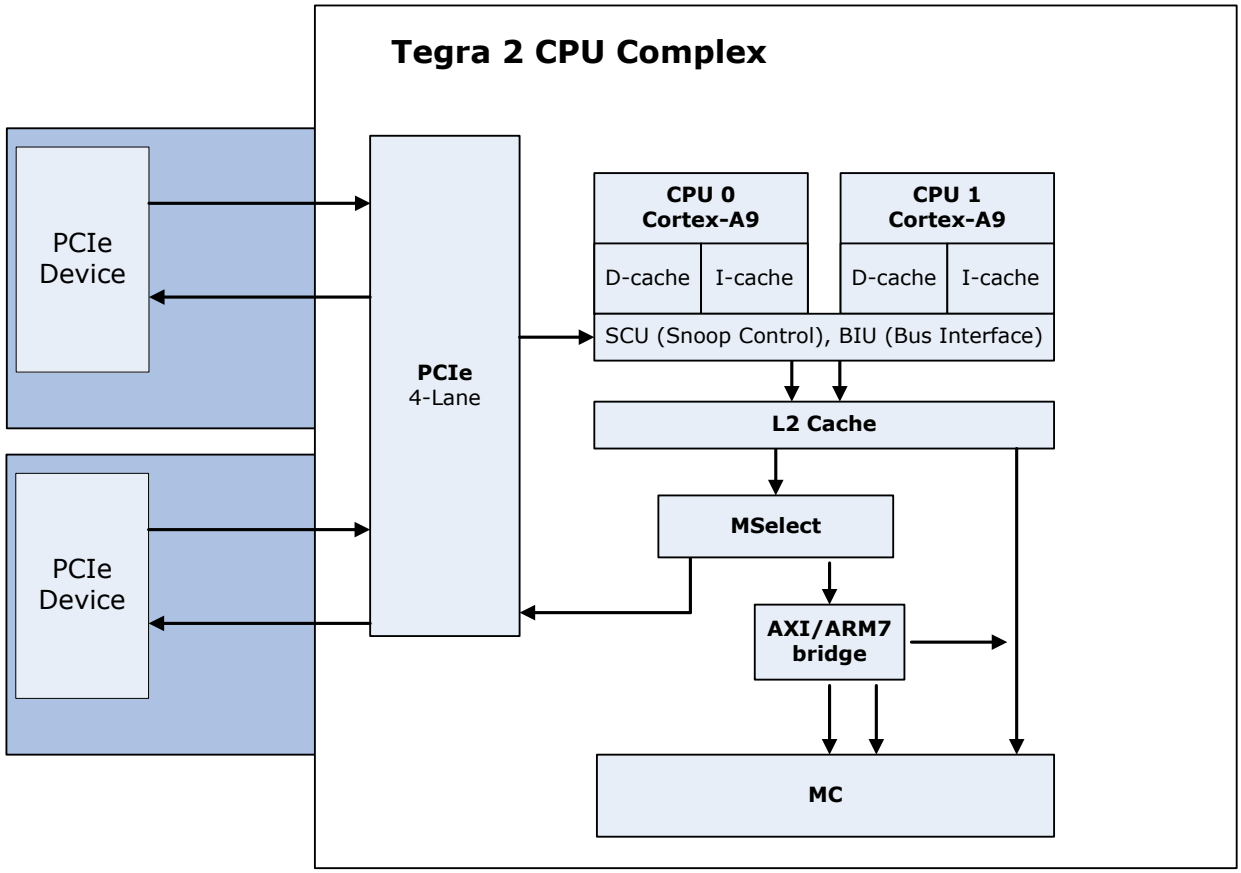
Bit	Reset	Description
5	X	DSI_PAD_CDN_CLK
4	X	DSI_PAD_CDP_CLK

Bit	Reset	Description
3:2	X	DSI_PAD_CDN
1:0	X	DSI_PAD_CDP

31.0 PCIE (TEGRA 250 ONLY)

The Tegra® 2 Processor incorporates a 4-lane PCIe bridge with support for both upstream and downstream FPCI interfaces that serves as the control path from Tegra 2 to the external PCIe device. The PCIe bridge enables a connection to one or two PCIe endpoints. When connected to two endpoints, the PCIe bridge requires the PCIe lanes to be split between two controllers; each controlling half of the PCIe lanes.

Figure 104 PCIe Bridge



31.1 Supported Configurations and Limitations

Tegra 2 supports the following lane/controller (number of lanes by number of controllers) configurations:

- 4x1
- 2x2

Tegra 2 requires relaxed ordering for responses to downstream requests (responses can pass writes). It is possible in some circumstances for PCIe transfers from an external bus masters (i.e. upstream transfers) to become blocked by a downstream read or non-posted write. The responses to these downstream requests are blocked by upstream posted writes only when PCIe strict ordering is imposed. It is therefore necessary to never impose strict ordering that would block a response to a downstream NPW/read request and always set the relaxed ordering bit to 1. Only devices that are capable of relaxed ordering may be used with Tegra 2 devices.

Strict ordering is still imposed on read and write requests traveling from the same source (reads push writes). This applies to both downstream and upstream requests.

31.2 Registers

31.2.1 PCI Compatible Configuration Registers

31.2.1.1 AP_PCIE2_RP_DEV_ID

This register implements the Device ID and Vendor ID registers as defined by the PCI 3.0 Specification.

Device ID and Vendor ID Register

Offset: 000h | Read/Write: RO

Bit	Reset	Description
31:16	None	**AP_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. BF0h **DEVICE_ID_TMS0_CTLR0_W4** BF1h **DEVICE_ID_TMS0_CTLR1_W2**
15:0	10DEh	**AP_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)

31.2.1.2 AP_PCIE2_RP_DEV_CTRL

Command and Status Register

Offset: 004h | Read/Write: R/W

Bit	R/W	Reset	Description
31	RW1C	0h	**AP_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	**AP_PCIE2_RP_DEV_CTRL_SIGNALED_SERR**: This bit is set when the Primary side device sends an ERR_FATAL or ERR_NONFATAL message, and the SERR Enable bit is set. 0h **SIGNALED_SERR_NOT_ACTIVE** (default) 1h **SIGNALED_SERR_ACTIVE** 1h **SIGNALED_SERR_SET**
29	RW1C	0h	**AP_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**

Bit	R/W	Reset	Description
28	RW1C	0h	**AP_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	**AP_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	**AP_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	**AP_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	**AP_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	**AP_PCIE2_RP_DEV_CTRL_66MHZ**: Does not apply to PCI Express. Hardwired to 0. 0h **66MHZ_INCAPABLE** (default) 1h **66MHZ_CAPABLE**
20	R	1h	**AP_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**
19	R	0h	**AP_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
10	R/W	0h	**AP_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**

Bit	R/W	Reset	Description
9	R	0h	**APAP_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	**AP_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	**AP_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	**AP_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	**AP_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	**AP_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	**AP_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	**AP_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	**AP_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**
0	R/W	0h	**AP_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**

31.2.1.3 AP_PCIE2_RP_REV_CC

This register implements the Revision ID and Class Code registers as defined by the PCI 3.0 Specification.

Revision ID and Class Code Registers

Offset: 008h | Read/Write: RO

Bit	Reset	Description
31:8	604000h	**AP_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	A0h	**AP_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 will be changed every time there is an all layer change to the MCP (A, B, C ...). The Minor Revision will be updated through metal edits each time there is a metal revision changing the functionality of this block. A1h **REVISION_ID_VAL**

31.2.1.4 AP_PCIE2_RP_MISC_1

This register implements the Cache Line Size, Latency Timer, Header Type, and BIST registers as defined by the PCI 3.0 Specification.

Cache Line Size, Latency Timer, Header Type, and BIST Registers

Offset: 00ch | Read/Write: R/W

Bit	R/W	Reset	Description
31:24	R	0h	**AP_PCIE2_RP_MISC_1_BIST**: The BIST register field is optional and not used. Hardwired to 0. 0h **BIST_ZERO** (default)
23	R	0h	**AP_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**
22:16	R	1h	**AP_PCIE2_RP_MISC_1_HEADER_TYPE0**: The HEADER_TYPE bits identify the layout of the bytes 10h through 3FH in configuration space and also whether or not the device contains multiple functions. Bits 22:16 in this register specify the layout of bytes 10h through 3Fh. PCI-Express Root Port Devices always employ Type 1 headers, hence this register is hard-wired to 1h. 0h **HEADER_TYPE0_NON_BRIDGE** 1h **HEADER_TYPE0_P2P_BRIDGE** (default)
15:11	R	0h	**APAP_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

Bit	R/W	Reset	Description
7:0	R/W	0h	**APAP_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 3.0 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)

31.2.1.5 AP_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 hard-wired to 0.

Base Address Register 0

Offset: 010h | Read/Write: RO

Bit	Reset	Description
31:0	0h	**AP_PCIE2_RP_BAR_0_RESERVED**: This register is unused and hard-wired to 0. 0h **RESERVED_0** (default)

31.2.1.6 AP_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 hard-wired to 0.

Base Address Register 1

Offset: 014h | Read/Write: RO

Bit	Reset	Description
31:0	0h	**AP_PCIE2_RP_BAR_1_RESERVED**: This register is unused and hard-wired to 0. 0h **RESERVED_0** (default)

31.2.1.7 AP_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: 018h | Read/Write: R/W

Bit	R/W	Reset	Description
31:27	R	0h	**APAP_PCIE2_RP_BN_LT_SLATENCY_TIMER**: This register does not apply to PCI Express. Hardwired to 0. 0h **SLATENCY_TIMER_0_CLOCKS** (default)
26:24	R	0	Reserved

Bit	R/W	Reset	Description
23:16	R/W	0h	**APAP_PCIE2_RP_BN_LT_SUB_BUS_NUMBER**: Subordinate Bus Number register is used to record the bus number of the highest numbered PCI bus segment which is behind the bridge Configuration software programs the value in this register. The bridge uses this register in conjunction with the Secondary Bus Number register to determine when to respond to a Type 1 configuration transaction on the primary interface. 0h **SUB_BUS_NUMBER_0** (default) 1h **SUB_BUS_NUMBER_1** 2h **SUB_BUS_NUMBER_2** FFh **SUB_BUS_NUMBER_255**
15:8	R/W	0h	**APAP_PCIE2_RP_BN_LT_SEC_BUS_NUMBER**: The Secondary Bus Number register is used to record the bus number of the PCI bus segment to which the secondary interface of the bridge is connected. Configuration software programs the value in this register. The bridge uses this register to decode Type 1 configuration transactions on the primary interface and convert them to Type 0 accesses on the secondary interface. 0h **SEC_BUS_NUMBER_0** (default) 1h **SEC_BUS_NUMBER_1** 2h **SEC_BUS_NUMBER_2** FFh **SEC_BUS_NUMBER_255**
7:0	R/W	0h	**APAP_PCIE2_RP_BN_LT_PRI_BUS_NUMBER**: The Primary Bus Number register is used to record the bus number of the PCI bus segment to which the primary interface of the bridge is connected. Configuration software programs the value in this register. The bridge uses this register to decode Type 1 configuration transactions on the secondary interface that must be converted to Special Cycle transactions on the primary interface. 0h **PRI_BUS_NUMBER_0** (default)

31.2.1.8 AP_PCIE2_RP_IO_BL_SS

This register implements the I/O Base, I/O Limit, and Secondary Status registers as defined by the PCI-to-PCI Bridge Architecture Specification, Revision 1.2.

The I/O Base and I/O Limit registers define an address range that is used by the bridge to determine when to forward I/O transactions from one interface to the other.

The Secondary Status register is similar in function and bit definition to the Status register defined above; however, its bits reflect status conditions of the secondary interface (the Status register reflects the status conditions of the primary interface)

I/O Base, I/O Limit and Secondary Status Registers

Offset: 01ch | Read/Write: R/W

Bit	R/W	Reset	Description
31	RW1C	0h	**APAP_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	**APAP_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**

Bit	R/W	Reset	Description
29	RW1C	0h	**APAP_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	**APAP_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	**APAP_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	**APAP_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	R	0h	**APAP_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**
23	R	0h	**APAP_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	**APAP_PCIE2_RP_IO_BL_SS_66MHZ**: Does not apply to PCI Express. Hardwired to 0. 0h **66MHZ_INCAPABLE** (default) 1h **66MHZ_CAPABLE**
20:16	R	0	Reserved
15:12	R/W	0h	**APAP_PCIE2_RP_IO_BL_SS_IO_LIMIT**: This register corresponds to address bits AD[15:12] of the I/O limit. For the purpose of address decoding, the bridge assumes that the lower 12 address bits of the I/O limit are 0xFFF. The I/O Limit register can be programmed to a smaller value than the I/O Base register if there are no I/O addresses on the secondary side of the bridge. 0h **IO_LIMIT_ADDRESS_0** (default) 1h **IO_LIMIT_ADDRESS_256** 2h **IO_LIMIT_ADDRESS_512** Fh **IO_LIMIT_ADDRESS_64K**

Bit	R/W	Reset	Description
11:8	R	1h	**APAP_PCIE2_RP_IO_BL_SS_IO_LIMIT_SUPPORT**: Identifies the I/O addressing support. When 0h, the device supports only 16-bit addressing. When 1h, it supports 32-bit addressing, in which case the I/O Limit Upper 16 Bits register is used to extend the I/O limit to 32 bits. Hard-wired to 1h. 0h **IO_LIMIT_SUPPORT_16** 1h **IO_LIMIT_SUPPORT_32** (default)
7:4	R/W	0h	**APAP_PCIE2_RP_IO_BL_SS_IO_BASE**: This register corresponds to address bits AD[15:12] of the I/O base address. For the purpose of address decoding, the bridge assumes that the lower 12 address bits, AD[11:0], of the I/O base address are zero. 0h **IO_BASE_ADDRESS_0** (default) 1h **IO_BASE_ADDRESS_256** 2h **IO_BASE_ADDRESS_512** Fh **IO_BASE_ADDRESS_64K**
3:0	R	1h	**APAP_PCIE2_RP_IO_BL_SS_IO_BASE_SUPPORT**: Identifies the I/O addressing support. When 0h, the device only supports 16-bit addressing. When 1h, it supports 32-bit addressing, in which case the I/O Base Upper 16 Bits register is used to extend the I/O base address to 32 bits. Hard-wired to 1h. 0h **IO_BASE_SUPPORT_16** 1h **IO_BASE_SUPPORT_32** (default)

31.2.1.9 AP_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 registers define an address range that is used by the bridge to determine when to forward memory-mapped I/O transactions from one interface to the other.

Memory Base and Memory Limit Registers

Offset: 020h | Read/Write: R/W

Bit	R/W	Reset	Description
31:20	R/W	0h	**APAP_PCIE2_RP_MEM_BL_MEM_LIMIT**: This register corresponds to address bits AD[31:20] of the memory-mapped I/O limit. For the purpose of address decoding, the bridge assumes that the lower 20 address bits, AD[19:0], of the memory-mapped I/O limit are 0xFFFFF. The Memory Limit register must be programmed to a smaller value than the Memory Base register if there are no memory-mapped I/O addresses on the secondary side of the bridge. 0h **MEM_LIMIT_ADDRESS_0** (default) 1h **MEM_LIMIT_ADDRESS_1MEG** 2h **MEM_LIMIT_ADDRESS_2MEG** FFFh **MEM_LIMIT_ADDRESS_4GIG**
19:16	R	0	Reserved
15:4	R/W	1h	**APAP_PCIE2_RP_MEM_BL_MEM_BASE**: This register corresponds to address bits AD[31:20] of the memory-mapped I/O base address. For the purpose of address decoding, the bridge assumes that the lower 20 address bits, AD[19:0], of the memory-mapped I/O base address are zero. 0h **MEM_BASE_ADDRESS_0** 1h **MEM_BASE_ADDRESS_1MEG** (default) 2h **MEM_BASE_ADDRESS_2MEG** FFFh **MEM_BASE_ADDRESS_4GIG**
3:0	R	0	Reserved

31.2.1.10 AP_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 registers define an address range that is used by the bridge to determine when to forward prefetchable memory transactions from one interface to the other.

Prefetchable Memory Base and Prefetchable Memory Limit Registers

Offset: 024h | Read/Write: R/W

Bit	R/W	Reset	Description
31:20	R/W	0h	**APAP_PCIE2_RP_PRE_BL_PREFETCH_MEM_LIMIT**: This register corresponds to address bits AD[31:20] of the prefetchable memory limit. For the purpose of address decoding, the bridge assumes that the lower 20 address bits, AD[19:0], of the prefetchable memory limit are 0xFFFFF. The Prefetchable Memory Limit register must be programmed to a smaller value than the Prefetchable Memory Base register if there is no prefetchable memory on the secondary side of the bridge. 0h **PREFETCH_MEM_LIMIT_ADDRESS_0** (default) 1h **PREFETCH_MEM_LIMIT_ADDRESS_1MEG** 2h **PREFETCH_MEM_LIMIT_ADDRESS_2MEG** FFFh **PREFETCH_MEM_LIMIT_ADDRESS_4GIG**
19:16	R	1h	**APAP_PCIE2_RP_PRE_BL_L64BIT**: Identifies the prefetchable memory addressing support. When 0h, the device supports only 32-bit addressing. When 1h, it supports 64-bit addressing, in which case the Prefetchable Limit Upper 32 Bits register is used to extend the prefetchable memory limit to 64 bits. Hard-wired to 1h. 1h **L64BIT_YES** (default)
15:4	R/W	1h	**APAP_PCIE2_RP_PRE_BL_PREFETCH_MEM_BASE**: This register corresponds to address bits AD[31:20] of the prefetchable memory base address. For the purpose of address decoding, the bridge assumes that the lower 20 address bits, AD[19:0], of the Prefetchable memory base address are zero. 0h **PREFETCH_MEM_BASE_ADDRESS_0** 1h **PREFETCH_MEM_BASE_ADDRESS_1MEG** (default) 2h **PREFETCH_MEM_BASE_ADDRESS_2MEG** FFFh **PREFETCH_MEM_BASE_ADDRESS_4GIG**
3:0	R	1h	**APAP_PCIE2_RP_PRE_BL_B64BIT**: Identifies the prefetchable memory addressing support. When 0h, the device supports only 32-bit addressing. When 1h, it supports 64-bit addressing, in which case the Prefetchable Base Upper 32 Bits register is used to extend the prefetchable memory base address to 64 bits. Hard-wired to 1h. 1h **B64BIT_YES** (default)

31.2.1.11 AP_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: 028h | Read/Write: R/W

Bit	Reset	Description
31:0	0h	**AP_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)

31.2.1.12 AP_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: 02ch | Read/Write: R/W

Bit	Reset	Description
31:0	0h	**AP_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)

31.2.1.13 AP_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 Registers

Offset: 030h | Read/Write: R/W

Bit	Reset	Description
31:16	0h	**AP_PCIE2_RP_IO_BL_U16_LIMIT_UPPER_BITS**: This register specifies the upper 16 bits, corresponding to AD[31::16], of the 32-bit I/O limit. 0h **LIMIT_UPPER_BITS_0** (default)
15:0	0h	**AP_PCIE2_RP_IO_BL_U16_BASE_UPPER_BITS**: This register specifies the upper 16 bits, corresponding to AD[31::16], of the 32-bit I/O base address. 0h **BASE_UPPER_BITS_0** (default)

31.2.1.14 AP_PCIE2_RP_CAP_PTR

This register implements the Capabilities Pointer register as defined by the PCI 3.0 Specification.

Capabilities Pointer

Offset: 034h | Read/Write: RO

Bit	Reset	Description
31:8	0	Reserved
7:0	40h	**AP_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)

31.2.1.15 AP_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 hard-wired to 0.

Expansion ROM Base Address Register

Offset: 038h | Read/Write: RO

Bit	Reset	Description
31:0	0h	**AP_PCIE2_RP_ROM_BA_RESERVED**: This register is unused and hard-wired to 0. 0h **RESERVED_0** (default)

31.2.1.16 AP_PCIE2_RP_INTR_BCR

This register implements the Interrupt Line, Interrupt Pin, and Bridge Control registers as defined by the PCI 3.0 Specification and PCI-to-PCI Bridge Architecture Specification, Revision 1.2.

Interrupt Line, Interrupt Pin, and Bridge Control Registers

Offset: 03ch | Read/Write: R/W

Bit	R/W	Reset	Description
31:28	R	0	Reserved
27	R	0h	**AP_PCIE2_RP_INTR_BCR_DIS_TIMER_SERR**: Does not apply to PCI Express. Hard-wired to 0. 0h **DIS_TIMER_SERR_DISABLED** (default) 1h **DIS_TIMER_SERR_ENABLED**
26	R	0h	**AP_PCIE2_RP_INTR_BCR_DIS_TIMER_STATUS**: Does not apply to PCI Express. Hard-wired to 0. 0h **DIS_TIMER_STATUS_NOT_ACTIVE** (default) 1h **DIS_TIMER_STATUS_ACTIVE**
25	R	0h	**AP_PCIE2_RP_INTR_BCR_SECONDARY_DIS_TIMER**: Does not apply to PCI Express. Hard-wired to 0. 0h **SECONDARY_DIS_TIMER_LONG** (default) 1h **SECONDARY_DIS_TIMER_SHORT**
24	R	0h	**AP_PCIE2_RP_INTR_BCR_PRIMARY_DIS_TIMER**: Does not apply to PCI Express. Hard-wired to 0. 0h **PRIMARY_DIS_TIMER_LONG** (default) 1h **PRIMARY_DIS_TIMER_SHORT**
23	R	0h	**AP_PCIE2_RP_INTR_BCR_FAST_B2B**: Does not apply to PCI Express. Hard-wired to 0. 0h **FAST_B2B_DISABLED** (default) 1h **FAST_B2B_ENABLED**
22	R/W	0h	**AP_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	**AP_PCIE2_RP_INTR_BCR_MABORT**: Does not apply to PCI Express. Hard-wired to 0. 0h **MABORT_DISABLED** (default) 1h **MABORT_ENABLED**

Bit	R/W	Reset	Description
20	R/W	0h	**AP_PCIE2_RP_INTR_BCR_VGA_16BITIO**: When Enabled, allows full 16-bit decode of IO address range for VGA. 0h **VGA_16BITIO_DISABLED** (default) 1h **VGA_16BITIO_ENABLED**
19	R/W	0h	**AP_PCIE2_RP_INTR_BCR_VGA_ADDRESS**: When enabled the P2P bridge will claim all of the legacy VGA addresses. Memory address range: 000A_0000 - 000B_FFFF IO address ranges: 3B0-3BB, 3C0-3DF (including all aliases if VGA_16BITIO is not set). 0h **VGA_ADDRESS_DISABLED** (default) 1h **VGA_ADDRESS_ENABLED**
18	R/W	0h	**AP_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	**AP_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	**AP_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	**AP_PCIE2_RP_INTR_BCR_INTR_PIN**: This register contains the interrupt pin the device (or device function) uses. A value of 1 corresponds to INTA#. A value of 2 corresponds to INTB#. A value of 3 corresponds to INTC#. A value of 4 corresponds to INTD#. Devices (or device functions) that do not use an interrupt pin must put a 0 in this register. This register is read-only. 0h **INTR_PIN_NONE** 1h **INTR_PIN_INTA** (default) 2h **INTR_PIN_INTB** 3h **INTR_PIN_INTC** 4h **INTR_PIN_INTD**
7:0	R/W	0h	**AP_PCIE2_RP_INTR_BCR_INTR_LINE**: This register contains the interrupt routing information. The register is read/write and must be implemented by any device (or device function) that uses an interrupt pin. POST software will write the routing information into this register as it initializes and configures the system. The value in this register tells which input of the system interrupt controller(s) the device's interrupt pin is connected to. Device drivers and operating systems can use this information to determine priority and vector information. INTR_LINE is written to 0xff (no connection) by software if the bridge does not use an interrupt pin. Some PCI BIOSes cannot handle aliased INTR_LINEs. Some PCI BIOSes cannot handle INTR_LINE initialized to 0xff. 0h **INTR_LINE_IRQ0** (default) 1h **INTR_LINE_IRQ1** Fh **INTR_LINE_IRQ15** FFh **INTR_LINE_UNKNOWN**

31.2.2 PCI Subsystem ID and Subsystem Vendor ID Capability Registers

31.2.2.1 AP_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: 040h | Read/Write: RO

Bit	Reset	Description
31:16	0	Reserved
15:8	48h	**AP_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	**AP_PCIE2_RP_SS_0_CAP_ID**: This read-only field is used to detect the presence of the SSID/SSVID registers in a PCI-to-PCI bridge. Hard-wired to 0Dh. Dh **CAP_ID_SS** (default)

31.2.2.2 AP_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: 044h | Read/Write: RO

Bit	Reset	Description
31:16	0h	**AP_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	AP_PCIE2_RP_SS_1_SSVID: The SSVID identifies the manufacturer of the add-in card or subsystem. The SSVID is assigned by PCI-SIG to insure uniqueness (the Vendor ID is used as the SSVID also). 10DEh SSVID_INIT (default)

31.2.3 PCI Power Management Capability Structure Registers

31.2.3.1 AP_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: 048h | Read/Write: RO

Bit	Reset	Description
31:27	1Fh	AP_PCIE2_RP_PM_0_PME_SUPPORT: This 5-bit field indicates the power states in which the function may assert PME#. A value of 0b for any bit indicates that the function is not capable of asserting the PME# signal while in that power state. bit(11) XXXX1b = PME# can be asserted from D0 bit(12) XXX1Xb = PME# can be asserted from D1 bit(13) XX1XXb = PME# can be asserted from D2 bit(14) X1XXXb = PME# can be asserted from D3hot bit(15) 1XXXXb = PME# can be asserted from D3cold Bits 31, 30, and 27 must be set to 1b for PCI-PCI Bridge structures representing Ports on Root Complexes/Switches to indicate that the Bridge will forward PME Messages. 1Fh PME_SUPPORT_YES (default) 0h PME_SUPPORT_NO
26	0h	**AP_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	**AP_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	AP_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	**AP_PCIE2_RP_PM_0_DEV_SPEC_INIT**: The Device Specific Initialization bit indicates whether special initialization of this function is required (beyond the standard PCI configuration header) before the generic class device driver is able to use it. Note that this bit is not used by some operating systems. Microsoft Windows and Windows NT, for instance, do not use this bit to determine whether to use D3. Instead, they use the driver's capabilities to determine this. A "1" indicates that the function requires a device specific initialization sequence following transition to the D0 uninitialized state. 0h **DEV_SPEC_INIT_NOT_NEEDED** (default) 1h **DEV_SPEC_INIT_NEEDED**
20	0	Reserved

Bit	Reset	Description
19	0h	**AP_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	**AP_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	**AP_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	**AP_PCIE2_RP_PM_0_CAP_ID**: This read-only field is used to detect the presence of the Power Management Capability registers in a PCI-to-PCI bridge. Hard-wired to 01h. 1h **CAP_ID_PM** (default)

31.2.3.2 AP_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: 04ch | Read/Write: R/W

Bit	R/W	Reset	Description
31:24	R	0h	**AP_PCIE2_RP_PM_1_PME_DATA**: The PME Data register is not implemented. Hard-wired to 0. 0h **PME_DATA_UNS** (default)
23	R	0h	**AP_PCIE2_RP_PM_1_PME_BPCC:** The bus power/clock control mechanism is not used. Hard-wired to 0. 0h **PME_BPCC_UNS** (default)
22	R	0h	**AP_PCIE2_RP_PM_1_PME_B2B3:** The bus power/clock control mechanism is not used. Therefore the B2/B3 support bit is hard-wired to 0. 0h **PME_B2B3_UNS** (default)
21:16	R	0	Reserved
15	RW1C	0h	**AP_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. 0h **PME_STATUS_NOT_ACTIVE** (default) 1h **PME_STATUS_ACTIVE** 1h **PME_STATUS_SET**
14:13	R	0h	**AP_PCIE2_RP_PM_1_PME_DATA_SCALE:** The PME Data register is not implemented. Hard-wired to 0. 0h **PME_DATA_SCALE_UNS** (default)

Bit	R/W	Reset	Description
12:9	R	0h	**AP_PCIE2_RP_PM_1_PME_DATA_SEL**: The PME Data register is not implemented. Hard-wired to 0. 0h **PME_DATA_SEL_UNS** (default)
8	R/W	0h	**AP_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. 0h PME_DISABLE (default) 1h PME_ENABLE
7:2	R	0	Reserved
1:0	R/W	0h	**AP_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**

31.2.4 PCI MSI Capability Structure Registers

31.2.4.1 AP_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: 050h | Read/Write: R/W

Bit	R/W	Reset	Description
31:24	R	0h	**AP_PCIE2_RP_MSI_CTRL_RSVD:** The RSVD field is reserved by the specification. 0h **RSVD_0** (default)
23	R	1h	**AP_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	**AP_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	**AP_PCIE2_RP_MSI_CTRL_MULT_CAP**: The MULT_CAP field determines the number of requested messages. It should always be a power of 2.

Bit	R/W	Reset	Description
			1h **MULT_CAP_CODE2** (default)
16	R/W	0h	**AP_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	**AP_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	**AP_PCIE2_RP_MSI_CTRL_CAP_ID:** This read-only field is used to detect the presence of the Message Signaled Interrupt Capability registers in a PCI-to-PCI bridge. Hard-wired to 05h. 5h **CAP_ID_MSI** (default)

31.2.4.2 AP_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: 054h | Read/Write: R/W

Bit	R/W	Reset	Description
31:2	R/W	0h	**AP_PCIE2_RP_MSI_LOW_ADDR_DWORD**: Bits 31:2 of the address. 0h **DWORD_0** (default)
1:0	R	0h	**AP_PCIE2_RP_MSI_LOW_ADDR_RSVD**: The address is always DWORD aligned, hence bits 1:0 of this register are hard-wired to 0. 0h **RSVD_0** (default)

31.2.4.3 AP_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: 058h | Read/Write: R/W

Bit	Reset	Description
31:0	0h	**AP_PCIE2_RP_MSI_UPPER_ADDR_DWORD**: Bits 63:32 of the address. 0h **DWORD_0** (default)

31.2.4.4 AP_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: 05ch | Read/Write: R/W

Bit	R/W	Reset	Description
31:16	R	0h	**AP_PCIE2_RP_MSI_DATA_RSVD**: 0h **RSVD_0** (default)
15:0	R/W	0h	**AP_PCIE2_RP_MSI_DATA_DATA**: The MSI message data. 0h **DATA_0** (default)

31.2.5 PCI Express Capability Structure Registers

31.2.5.1 AP_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: 080h | Read/Write: RO

Bit	Reset	Description
31:30	0	Reserved
29:25	0h	**AP_PCIE2_RP_PCI_EXPRESS_CAPABILITY_INTERRUPT_MESSAGE_NUMBER**: If this function is allocated more than one MSI interrupt number, this register is required to contain the offset between the base Message Data and the MSI Message that is generated when any of the status bits in either the Slot Status register or the Root Port Status register of this capability structure are set. Hardware is required to update this field so that it is correct if the number of the MSI Messages assigned to the device changes. 0h **INTERRUPT_MESSAGE_NUMBER_ZERO** (default)
24	1h	**AP_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	**AP_PCIE2_RP_PCI_EXPRESS_CAPABILITY_DEVICE_PORT_TYPE:** Indicates the type of PCI Express logical device. Defined encodings are: 0000b = PCI Express Endpoint device. 0001b = Legacy PCI Express Endpoint device. 0100b = Root Port of PCI Express Root Complex. 0101b = Upstream Port of PCI Express Switch. 0110b = Downstream Port of PCI Express Switch. 0111b = PCI Express-to-PCI/PCI-X Bridge. 1000b = PCI/PCI-X to PCI Express Bridge **4h DEVICE_PORT_TYPE_INIT (default)**

Bit	Reset	Description
19:16	2h	**AP_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	**AP_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	**AP_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)

31.2.5.2 AP_PCIE2_RP_DEVICE_CAPABILITY

The Device Capabilities register identifies PCI Express device specific capabilities.

Device Capabilities Register

Offset: 084h | Read/Write: RO

Bit	Reset	Description
31:28	0	Reserved
27:26	0h	**AP_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	**AP_PCIE2_RP_DEVICE_CAPABILITY_CAPTURED_SLOT_POWER_LIMIT_VALUE**: In combination with the Slot Power Limit Scale value, specifies the upper limit on power supplied by slot. Power limit (in Watts) is calculated by multiplying the value in this field by the value in the Slot Power Limit Scale field. This value is set by the Set_Slot_Power_Limit message or hardwired to 0000_0000b. The default value is 0000_0000b. 0h **CAPTURED_SLOT_POWER_LIMIT_VALUE_INIT** (default)
17:16	0	Reserved
15	1h	**AP_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 PCIE Base Spec 1.0a and later incorporated into PCIE Base Spec 1.1. 1h **ROLE_BASED_ERR_REPORTING_INIT** (default)
14	0h	**AP_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	**AP_PCIE2_RP_DEVICE_CAPABILITY_ATTENTION_INDICATOR_PRESENT**: This bit, when set, indicates that an Attention Indicator is implemented on the card or module. 0h **ATTENTION_INDICATOR_PRESENT_INIT** (default)

Bit	Reset	Description
12	0h	**AP_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	**AP_PCIE2_RP_DEVICE_CAPABILITY_ENDPOINT_L1_ACCEPTABLE_LATENCY:** This field indicates acceptable latency that an Endpoint can withstand due to the transition from L1 state to the L0 state. It is essentially an indirect measure of the Endpoint's internal buffering. • 000b -- Less than 1 us. • 001b -- 1 us to less than 2 us. • 010b -- 2 us to less than 4 us. • 011b -- 4 us to less than 8 us. • 100b -- 8 us to less than 16 us. • 101b -- 16 us to less than 32 us. • 110b -- 32 us-64 us. • 111b -- More than 64 us. 0h **ENDPOINT_L1_ACCEPTABLE_LATENCY_INIT** (default)
8:6	0h	**AP_PCIE2_RP_DEVICE_CAPABILITY_ENDPOINT_L0S_ACCEPTABLE_LATENCY**: This field indicates the acceptable total latency that an Endpoint can withstand due to the transition from L0s state to the L0 state. It is essentially an indirect measure of the Endpoint's internal buffering. • 000b -- Less than 64 ns. • 001b -- 64 ns to less than 128 ns. • 010b -- 128 ns to less than 256 ns. • 011b -- 256 ns to less than 512 ns. • 100b -- 512 ns to less than 1 us. • 101b -- 1 us to less than 2 us. • 110b -- 2 us-4us. • 111b -- More than 4 us. 0h **ENDPOINT_L0S_ACCEPTABLE_LATENCY_INIT** (default)
5	None	**AP_PCIE2_RP_DEVICE_CAPABILITY_EXTENDED_TAG_FIELD_SIZE**: This field indicates the maximum supported size of the Tag field. Defined encodings are: 0b = 5-bit Tag field supported. 1b = 8-bit Tag field supported.
4:3	0h	**AP_PCIE2_RP_DEVICE_CAPABILITY_PHANTOM_FUNCTIONS_SUPPORTED:** This field indicates the support for use of unclaimed function numbers to extend the number of outstanding transactions allowed by logically combining unclaimed function numbers (called Phantom Functions) with the Tag identifier. This field indicates the number of most significant bits of the function number portion of Requester ID that are logically combined with the Tag identifier. Defined encodings are: • 00b -- No function number bits used for Phantom Functions; device may implement all function numbers. • 01b -- First most significant bit of the function number in Requester ID used for Phantom Functions; device may implement functions 0-3. Functions 0, 1, 2 and 3 may claim functions 4, 5, 6, and 7 as Phantom Functions respectively. • 10b -- First two most significant bits of the function number in Requestor ID used for Phantom Functions; device may implement functions 0-1. Function 0 may claim functions 2, 4 and 6 as Phantom Functions, function 1 may claim functions 3, 5 and 7 as Phantom Functions. • 11b -- All three bits of function number in Requester ID used for Phantom Functions; device must be a single function 0 device that may claim all other functions as Phantom Functions. **Note:** The Phantom Function support for the device must be enabled by the corresponding control field in the Device Control Register. 0h PHANTOM_FUNCTIONS_SUPPORTED_INIT (default)

Bit	Reset	Description
2:0	0h	**AP_PCIE2_RP_DEVICE_CAPABILITY_MAX_PAYLOAD_SIZE:** This field indicates the maximum payload size that the device can support for TLP's. • 000b --128B max payload size • 001b -- 256B max payload size • 010b -- 512B max payload size • 011b -- 1024B max payload size • 100b -- 2048B max payload size • 101b -- 4096B max payload size • 110b -- Reserved • 111b -- Reserved 0h MAX_PAYLOAD_SIZE_INIT (default)

31.2.5.3 AP_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: 088h | Read/Write: R/W

Bit	R/W	Reset	Description
31:22	R	0	Reserved
21	R	0h	**AP_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	**AP_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	**AP_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. 0h **UNSUPP_REQUEST_DETECTED_INIT (default)** **1h UNSUPP_REQUEST_DETECTED_SET**
18	RW1C	0h	**AP_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	**AP_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. 0h **NON_FATAL_ERROR_DETECTED_INIT** (default) 1h **NON_FATAL_ERROR_DETECTED_SET**

Bit	R/W	Reset	Description
16	RW1C	0h	**AP_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. 0h **CORR_ERROR_DETECTED_INIT** (default) 1h **CORR_ERROR_DETECTED_SET**
15	R	0	Reserved
14:12	R/W	2h	**AP_PCIE2_RP_DEVICE_CONTROL_STATUS_MAX_READ_REQUEST_SIZE:** This field sets the maximum Read Request size for the Device as a Requester. The Device must not generate read requests with size exceeding the set value. Defined encodings for this field are: • 000b -- 128B max payload size • 001b -- 256B max payload size • 010b -- 512B max payload size • 011b -- 1024B max payload size • 100b -- 2048B max payload size • 101b -- 4096B max payload size • 110b -- Reserved • 111b – Reserved Devices that do not generate Read Requests larger than 128 bytes are permitted to implement this field as Read Only (RO) with a value of 000b. 2h **MAX_READ_REQUEST_SIZE_INIT** (default)
11	R/W	1h	**AP_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	**AP_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. 0h **AUXILLARY_POWER_PM_ENABLE_INIT** (default)
9	R	0h	**AP_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	**AP_PCIE2_RP_DEVICE_CONTROL_STATUS_EXTENDED_TAG_FIELD_ENABLE:** When set, this bit enables a device to use an 8-bit Tag field as a requester. If the bit is cleared, the device is restricted to a 5-bit Tag field. Devices that do not implement this capability hardwire this bit to 0. 0h **EXTENDED_TAG_FIELD_ENABLE_INIT** (default)

Bit	R/W	Reset	Description
7:5	R/W	0h	**AP_PCIE2_RP_DEVICE_CONTROL_STATUS_MAX_PAYLOAD_SIZE:** This field sets the maximum TLP payload size for the device. As a receiver, the device must handle TLP's as large as the set value; as transmitter, the device must not generate TLPs exceeding the set value. Permissible values that can be programmed are indicated by the Max_Payload_Size supported in the Device Capabilities register. Defined encodings for this field are: • 000b -- 128B max payload size • 001b -- 256B max payload size • 010b -- 512B max payload size • 011b -- 1024B max payload size • 100b -- 2048B max payload size • 101b -- 4096B max payload size • 110b -- Reserved • 111b -- Reserved 0h **MAX_PAYLOAD_SIZE_INIT** (default)
4	R/W	1h	**AP_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	**AP_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	**AP_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	**AP_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	**AP_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)

31.2.5.4 AP_PCIE2_RP_LINK_CAPABILITIES

The Link Capabilities Register identifies PCI Express Link specific capabilities.

Link Capabilities Register

Offset: 08ch | Read/Write: RO

Bit	Reset	Description
31:24	None	**AP_PCIE2_RP_LINK_CAPABILITIES_PORT_NUMBER**: This field indicates the PCI Express port number for the given PCI Express Link.
23:22	None	**AP_PCIE2_RP_LINK_CAPABILITIES_RESERVED**: Reserved.

Bit	Reset	Description
21	0	Reserved
20	1h	**AP_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	**AP_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	**AP_PCIE2_RP_LINK_CAPABILITIES_CLOCK_PM**: 0h **CLOCK_PM_INIT** (default)
17:15	2h	**AP_PCIE2_RP_LINK_CAPABILITIES_L1_EXIT_LATENCY:** This field indicates the L1 exit latency for the given PCI Express Link. The value reported indicates the lenght of time this Port requires to complete transition from L1 to L0. Defined encodings are: • 000b -- Less than 1 us. • 001b -- 1 us to less than 2 us. • 010b -- 2 us to less than 4 us. • 011b -- 4 us to less than 8 us. • 100b -- 8 us to less than 16 us. • 101b -- 16 us to less than 32 us. • 110b -- 32 us-64 us. • 111b -- More than 64 us. 2h L1_EXIT_LATENCY_INIT (default)
14:12	3h	**AP_PCIE2_RP_LINK_CAPABILITIES_L0S_EXIT_LATENCY:** This field indicates the L0s exit latency for the given PCI Express Link. The value reported indicates the length of time this Port requires to complete transition from L0s state to L0. Defined encodings are: • 000b -- Less than 64 ns. • 001b -- 64 ns to less than 128 ns. • 010b -- 128 ns to less than 256 ns. • 011b -- 256 ns to less than 512 ns. • 100b -- 512 ns to less than 1 us. • 101b -- 1 us to less than 2 us. • 110b -- 2 us-4us. • 111b -- Reserved. 3h L0S_EXIT_LATENCY_INIT (default)
11:10	11h	**AP_PCIE2_RP_LINK_CAPABILITIES_ACTIVE_STATE_LINK_PM_SUPPORT**: This field indicates the level of active state power management supported on the given PCI Express Link. Defined encodings are: • 00b -- Reserved • 01b -- L0s Entry Supported • 10b -- Reserved • 11b -- L0s and L1 Supported 11h ACTIVE_STATE_LINK_PM_SUPPORT_INIT (default)

Bit	Reset	Description
9:4	None	**AP_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	**AP_PCIE2_RP_LINK_CAPABILITIES_MAX_LINK_SPEED:** This field indicates the maximum Link speed of the given PCI Express Link. Defined encodings are: 0001b = 2.5 Gb/s Link All other encodings are reserved. 1h MAX_LINK_SPEED_INIT (default)

31.2.5.5 AP_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: 090h | Read/Write: R/W

Bit	R/W	Reset	Description
31:30	R	0	Reserved
29	R	None	**AP_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	**AP_PCIE2_RP_LINK_CONTROL_STATUS_SLOT_CLOCK_CONFIG:** This bit indicates that the component uses the same physical refreference 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	**AP_PCIE2_RP_LINK_CONTROL_STATUS_LINK_TRAINING:** This read-only bit indicates that Link training is in progress; hardware clears this bit once training is complete. Note: This field is not applicable and reserved for endpoint devices and Upstream Ports of Switches.
26	R	None	**AP_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	**AP_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

Bit	R/W	Reset	Description
19:16	R	1h	**AP_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)
15:8	R	0	Reserved
7	R/W	0h	**AP_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	**AP_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	**AP_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	**AP_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	**AP_PCIE2_RP_LINK_CONTROL_STATUS_READ_COMPLETION_BOUNDARY:** Encodings are: 0b = 64 byte 1b = 128 byte This field is hardwired for a Root Port and returns its RCB support capabilities. Devices that do not implement this feature must hardwire the field to 0b. This field is R/W for devices other than Root Ports. 0h READ_COMPLETION_BOUNDARY_INIT (default)
2	R	0	Reserved
1:0	R/W	0h	**AP_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)

31.2.5.6 AP_PCIE2_RP_SLOT_CAPABILITIES

The Slot Capabilities register identifies PCI Express slot specific capabilities.

Slot Capabilities Register

Offset: 094h | Read/Write: RO

Bit	Reset	Description
31:19	None	**AP_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	**AP_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	**AP_PCIE2_RP_SLOT_CAPABILITIES_ELECTROMECHANICAL_INTERLOCK_PRESENT:** Indicates that the ElectroMechanical Interlock mechanism is implemented. 0h ELECTROMECHANICAL_INTERLOCK_PRESENT_NO (default)
16:15	None	**AP_PCIE2_RP_SLOT_CAPABILITIES_SLOT_POWER_LIMIT_SCALE:** This field Specifies the scale used for the Slot Power Limit Value. Range of values: 00b = 1.0x 01b = 0.1x 10b = 0.01x 11b = 0.001x This register must be implemented if the Slot Implemented bit is set. The default value prior to hardware/firmware initialization is 00b.
14:7	None	**AP_PCIE2_RP_SLOT_CAPABILITIES_SLOT_POWER_LIMIT_VALUE:** In combination with the Slot Power Limit Scale value, this field specifies the upper limit on power supplied by slot. Power limit (in Watts) is calculated by multiplying the value in this field by the value in the Slot Power Limit Scale field. This register must be implemented if the Slot Implemented bit is set. The default value prior to hardware/firmware initialization is 0000_0000b.
6	0h	**AP_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	**AP_PCIE2_RP_SLOT_CAPABILITIES_HOT_PLUG_SURPRISE:** This bit when set indicates that a device present in this slot might be removed from the system without any prior notification. 0h HOT_PLUG_SURPRISE_NO (default)
4	0h	**AP_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	**AP_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	**AP_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	**AP_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	**AP_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)

31.2.5.7 AP_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: 098h | Read/Write: R/W

Bit	R/W	Reset	Description
31:25	R	0	Reserved
24	RW1C	0h	**AP_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	**AP_PCIE2_RP_SLOT_CONTROL_STATUS_ELECTROMECHANICAL_INTERLOCK_STATE:** This bit when set physically locks the add-in card which the user is trying to remove, till software releases it by resetting the bit. Indicates the current state of the interlock, ie whether the card is electro-mechanically engaged or disengaged 0b : disengaged 1b : engaged Software reads this bit to determine what state the card is in 1h ELECTROMECHANICAL_INTERLOCK_STATE_YES
22	R	None	**AP_PCIE2_RP_SLOT_CONTROL_STATUS_PRESENCE_DETECT_STATE:** This bit indicates the presence of a card in the slot. The bit reflects the Presence Detect status determined via an in-band mechanism or via the Present Detect pins as defined in the PCI Express Card Electromechanical Specification. Defined encodings are: 0b = Slot Empty. 1b = Card Present in slot. This register is required to be implemented on all Switch devices and Root Ports. The presence detect field for Switch devices or Root Ports not connected to slots should be hardwired to 1. This register is required if a slot is implemented. 1h PRESENCE_DETECT_STATE_YES
21	R	0h	**AP_PCIE2_RP_SLOT_CONTROL_STATUS_MRL_SENSOR_STATE:** This register reports the status of the MRL sensor if it is implemented. Defined encodings are: 0b = MRL Closed. 1b = MRL Open. 0h MRL_SENSOR_STATE_INIT (default)
20	RW1C	0h	**AP_PCIE2_RP_SLOT_CONTROL_STATUS_COMMAND_COMPLETED:** This bit is set when the hot plug controller completes an issued command. 0h COMMAND_COMPLETED_INIT (default) 1h COMMAND_COMPLETED_SET
19	RW1C	0h	**AP_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	**AP_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	**AP_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	**AP_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

Bit	R/W	Reset	Description
15:13	R	0	Reserved
12	R/W	0h	**AP_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	**AP_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	**AP_PCIE2_RP_SLOT_CONTROL_STATUS_POWER_CONTROLLER_CONTROL:** When read this register returns the current state of the Power applied to the slot; when written sets the power state of the slot per the defined encodings. 0b = Power On. 1b = Power Off. 0h POWER_CONTROLLER_CONTROL_INIT (default)
9:8	R/W	1h	**AP_PCIE2_RP_SLOT_CONTROL_STATUS_POWER_INDICATOR_CONTROL:** Reads to this register return the current state of the Power Indicator; writes to this register set the Power Indicator. 00b = Reserved 01b = On 10b = Blink 11b = Off Writes to this register also cause the Port to send the appropriate POWER_INDICATOR_* messages. 1h POWER_INDICATOR_CONTROL_INIT (default)
7:6	R/W	3h	**AP_PCIE2_RP_SLOT_CONTROL_STATUS_ATTN_INDICATOR_CONTROL:** Reads to this register return the current state of the Attention Indicator; writes to this register set the Attention Indicator. Defined encodings are: 00b = Reserved 01b = On 10b = Blink 11b = Off Writes to this register also cause the Port to send the appropriate ATTENTION_INDICATOR_* messages. 3h ATTN_INDICATOR_CONTROL_INIT (default)
5	R/W	0h	**AP_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	**AP_PCIE2_RP_SLOT_CONTROL_STATUS_COMMAND_COMPLETED_INTERRUPT_ENABLE:** This bit when set enables the generation of hot plug interrupt when a command is completed by the Hot plug controller. 0h COMMAND_COMPLETED_INTERRUPT_ENABLE_INIT (default)
3	R/W	0h	**AP_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	**AP_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	**AP_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	**AP_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)

31.2.5.8 AP_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: 09ch | Read/Write: R/W

Bit	R/W	Reset	Description
31:4	R	0	Reserved
3	R/W	0h	**AP_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	**AP_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	**AP_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	**AP_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)

31.2.5.9 AP_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: 0A0h | Read/Write: R/W

Bit	R/W	Reset	Description
31:18	R	0	Reserved
17	R	None	**AP_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	**AP_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**

Bit	R/W	Reset	Description
15:0	R	None	**AP_PCIE2_RP_RSR_REQID**: The REQID field indicates the PCI requester ID of the last PME requestor.

31.2.5.10 AP_PCIE2_RP_DEVICE_CAPABILITIES_2

Offset: 0A4h | Read/Write: RO

Bit	Reset	Description
31:5	0h	**AP_PCIE2_RP_DEVICE_CAPABILITIES_2_RESERVED**: Unused bits in this register. 0h **RESERVED_0** (default)
4	1h	**AP_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	**AP_PCIE2_RP_DEVICE_CAPABILITIES_2_CPL_TO_RANGES_SUP**: Completion timeout ranges supported by the rootport. Ranges A and B are currently supported by the PCIE2 design. 3h **CPL_TO_RANGES_SUP_0** (default)

31.2.5.11 AP_PCIE2_RP_DEVICE_CONTROL_STATUS_2

This register is primarily used for Rootport Completion Timeout values. Bits 31:5 are reserved.

Offset: 0A8h | Read/Write: R/W

Bit	Reset	Description
31:5	0h	**AP_PCIE2_RP_DEVICE_CONTROL_STATUS_2_RESERVED:** Unused bits in this register. **0h RESERVED_0 (default)**
4	0h	**AP_PCIE2_RP_DEVICE_CONTROL_STATUS_2_CPL_TO_DISABLE:** Disables completion timeout, making the rootport always wait endlessly for a completion to a request. **0h CPL_TO_DISABLE_DEFAULT (default)**
3:0	0h	**AP_PCIE2_RP_DEVICE_CONTROL_STATUS_2_CPL_TO_VALUE:** This field determines the Range and hence the timeout values for Completions expected from the endpoint. Range A 50 Us to 100 ms (80 Us in design) 1 ms to 10 ms (5 ms in design) Range B 16 ms to 55 ms (40 ms in design) 65 ms to 210 ms (140 ms in design) Default 50 Us to 50 ms (10 ms in design) 1h **CPL_TO_VALUE_RANGE_A_LO** 2h **CPL_TO_VALUE_RANGE_A_HI** 5h **CPL_TO_VALUE_RANGE_B_LO** 6h **CPL_TO_VALUE_RANGE_B_HI** 0h **CPL_TO_VALUE_DEFAULT** (default)

31.2.5.12 AP_PCIE2_RP_LINK_CAPABILITIES_2

This is a placeholder register and is hardwired to 0. There are no capabilities that require this register.

Offset: 0Ach | Read/Write: RO

Bit	Reset	Description
31:0	0h	**AP_PCIE2_RP_LINK_CAPABILITIES_2_BITS**: 0h **BITS_0**

31.2.5.13 AP_PCIE2_RP_LINK_CONTROL_STATUS_2

Offset: 0B0h | Read/Write: R/W

Bit	R/W	Reset	Description
31:17	R	0h	**AP_PCIE2_RP_LINK_CONTROL_STATUS_2_RESERVED_STATUS**: 0h **RESERVED_STATUS_DEFAULT** (default)
16	R	None	**AP_PCIE2_RP_LINK_CONTROL_STATUS_2_CURRENT_DEEMPHASIS_LEVEL**: 1h **CURRENT_DEEMPHASIS_LEVEL_3P5** 0h **CURRENT_DEEMPHASIS_LEVEL_6**
15:13	R	0h	**AP_PCIE2_RP_LINK_CONTROL_STATUS_2_RESERVED_CONTROL**: 0h **RESERVED_CONTROL_DEFAULT** (default)
12	R/W	0h	**AP_PCIE2_RP_LINK_CONTROL_STATUS_2_COMPLIANCE_DEEMPHASIS**: 0h **COMPLIANCE_DEEMPHASIS_INIT** (default)
11	R/W	0h	**AP_PCIE2_RP_LINK_CONTROL_STATUS_2_COMPLIANCE_SOS**: 0h **COMPLIANCE_SOS_INIT** (default)
10	R/W	0h	**AP_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. 0h **ENTER_MODIFIED_COMPLIANCE_INIT** (default)
9:7	R/W	0h	**AP_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. 0h **TRANSMIT_MARGIN_INIT** (default)
6	R	None	**AP_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	**AP_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**

Bit	R/W	Reset	Description
4	R/W	0h	**AP_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. 0h **ENTER_COMPLIANCE_INIT** (default)
3:0	R/W	2h	**AP_PCIE2_RP_LINK_CONTROL_STATUS_2_TARGET_LINK_SPEED**: This field sets an upper limit on the link operational speed by restricting the values advertised in training sequences. Defined encodings: 0001b 2.5Gb/s (Gen1) 0010b 5Gb/s (Gen2) Other encodings are reserved. Default value is 0. 0h **TARGET_LINK_SPEED_GEN2_DIS** 1h **TARGET_LINK_SPEED_2P5** 2h **TARGET_LINK_SPEED_5P0** (default)

31.2.5.14 AP_PCIE2_RP_SLOT_CAPABILITIES_2

This is a placeholder register and is hardwired to 0. There are no capabilities that require this register.

Offset: 0B4h | Read/Write: RO

Bit	Reset	Description
31:0	0h	**AP_PCIE2_RP_SLOT_CAPABILITIES_2_BITS**: 0h **BITS_0**

31.2.5.15 AP_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: 0B8h | Read/Write: RO

Bit	Reset	Description
31:0	0h	**AP_PCIE2_RP_SLOT_CONTROL_STATUS_2_BITS**: 0h **BITS_0**

Notice

Trademarks

Copyright