Chapter 44
Multi Mode DDR Controller (MMDC)

44.1 Overview

MMDC is a multi-mode DDR controller that supports DDR3/DDR3L x16/x32/x64 and LPDDR2 two channel x16/x32 memory types. MMDC is configurable, high performance, and optimized.

The following figure shows the MMDC block diagram.

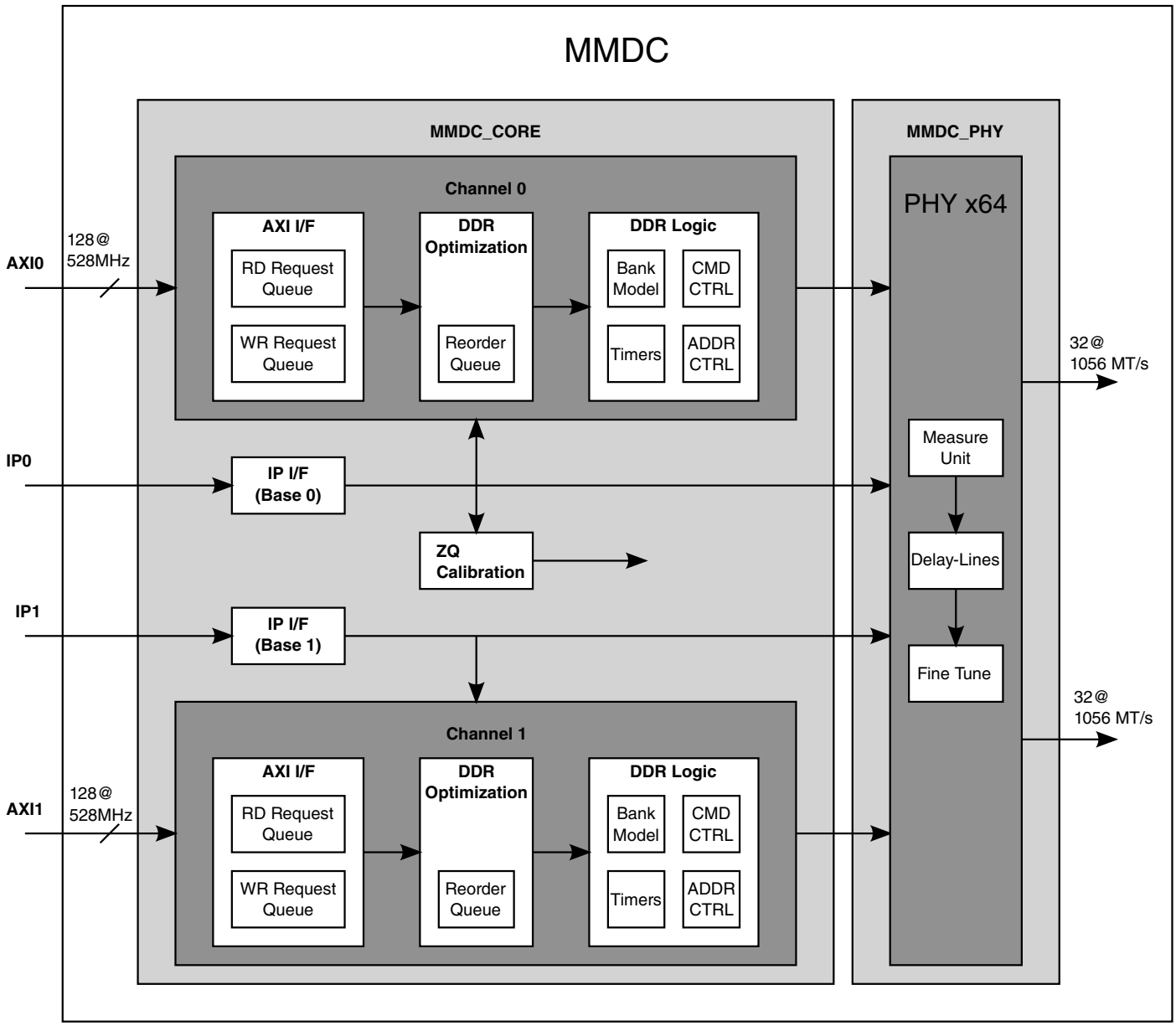


Figure 44-1. MMDC block diagram

MMDC consists of a core (MMDC_CORE) and PHY (MMDC_PHY).

- The core is responsible for communication with the system through an AXI interface, DDR command generation, DDR command optimizations, and a read/write data path.
- The PHY is responsible for the timing adjustment; it uses special calibration mechanisms to ensure data capture margin at a clock rate of up to 528 MHz.

NOTE

The core is composed of two channels, but both channels are only active in LPDDR2 mode. If DDR3 mode is selected,

channel1 is not activated and the MMDC communicates with the system through AXI port0.

The internal memory map(configuration registers) of the MMDC can be configured through two IP channels (IP0 and IP1). IP channel 0 is associated with 021B_0000h–021B_08C0h and IP channel1 is associated with 021B_4000h–021B_48C0h.

44.1.1 MMDC feature summary

The table found here summarizes the MMDC features.

Table 44-1. MMDC feature summary

Feature	Details
DDR standards	• LV-DDR3, DDR3 x16, x32, x64 (includes SODIMM) • LPDDR2 2-channels x16, x32 • Does not support LPDDR1MDDR or DDR2

DDR interface	• x16, x32, x64 data bus width
	• Density per DDR device of 256 Mbits–8 Gbits with the following column and row combinations:
	  • Column size of 8–12 bits
	  • Row size of 11–16 bits
	• Two chip selects per channel
	• Up to 4 Gbytes of address space with configurable partitioning between CS0 and CS1 (for LPDDR2 2ch x32 up to 2 Gbytes per channel)
	• Supports burst length of 8 (aligned) for DDR3
	• Supports burst length of 4 for LPDDR2

DDR performance	• MMDC running at up to 528 MHz (1056MT/s), see CCM block for actual clock frequencies supported. • Supports Real-Time priority by means of QoS sideband priority signals from the chip to enable various priority levels in the re-ordering mechanism: real-time, latency sensitive, normal priority. • Page hit/page miss optimizations • Consecutive read/write access optimizations • Supports deep read and write request queues to enable bank prediction. • Drives back the critical word in a read transaction as soon as it is received by the DDR device (does not wait until the whole data phase has been completed). • Keeps tracking of open memory pages • Supports bank interleaving • Special optimization in case of non-aligned wrap accesses in DDR3 mode (burst length 8) **NOTE:** Due to reordering and optimization mechanisms (per different AXI Identifier (ID)), the transactions towards the DDR device may be driven in a different ID order than was received by the AXI master. In a similar fashion, the write response, read response or read data may be driven to the AXI master in a different ID order.
AXI interface	• AXI bus compliant • Supports bus transfers of 8, 16, 32, 64 and 128 bits (single accesses and bursts) running at 528 MHz. • Supports AXI bursts length of up to 16 • Supports burst types of WRAP, INCR and FIXED • Supports 16 bits AXI ID

Table continues on the next page...

Table 44-1. MMDC feature summary (continued)

Feature	Details

	• Write data interleave depth is 1 (no support for Write Data Interleave)
	• Supports write data before address
	• Supports buffered/non-buffered accesses (AWCACHE[0] = 0b means a non-bufferable access and AWCACHE[0] = 1b means a bufferable access). The rest of the CACHE options are not supported
	  • To keep data access coherency between write and read access of the same master, the response signal is sent as follows:
	  • Bufferable write access—BRESP will be sent when last data of the access has entered the MMDC.
	  • Non-bufferable write access—BRESP will be sent when the data was physically written into the external memory device.
	• Supports four exclusive monitors per configurable ID for only a single access with a size of up to 64 bits
	• Supports AXI responses as follows:
	  • Okay in case the access has been successful or exclusive access failure
	  • Slave error in case of security violation
	  • Exclusive okay in case the read or the write portion of an exclusive access has been successful

DDR calibration and delay-lines.	• Supports various calibration processes which can be performed either automatically (hardware) or manually (software) towards either CS0 or CS1. (At the end of the process the delay-lines will work with one set of results.) The following calibration processes are supported:
	  • ZQ calibration for external DDR device (in DDR3 through ZQ calibration command and in LPDDR2 through MRW command)
	  • Can be handled automatically for ZQ Short (periodically) and ZQ Long (at exit from self-refresh)
	  • Can be handled manually at ZQ INIT
	  • ZQ calibration for i.MX DDR I/O pads for calibrating the DDR driving strength
	  • The sequence can be handled automatically by hardware
	  • The sequence can be handled step by step manually by software
	  • Read data calibration. Adjustment of read DQS with read data byte.
	  • Read DQS gating calibration for DDR3 only. Adjustment of DQS gate with read preamble window.
	  • Write data calibration. Adjustment of write DQS with write data byte.
	  • Write leveling calibration. Adjustment of write DQS with CK (DDR differential clock).
	  • Read fine tuning. Adjustment of up to 7 delay-line units for each read data bit.
	  • Write fine tuning. Adjustment of up to 3 delay-line units for each read data bit.
	  • Periodic delay-line measurement for keeping its accuracy during refresh interval.
	  • Additional fine tuning delay lines to adjust DDR clock delay, DDR clock duty cycle, DQS duty cycle.

Power saving	• Support of dynamic voltage, frequency change and self-refresh mode entry through hardware and software negotiation with the system (request/acknowledge handshake)
	  • Upon hardware or software self-refresh request assertion, further AXI requests are blocked (even before the assertion of the acknowledge).
	  • During self-refresh mode the system may deassert the operating clock of the MMDC for power saving.
	  • During self-refresh mode the clock (CK) that is driven to the DDR device will be gated for power saving.
	• Supports automatic self-refresh and power down entry and exit
	  • In automatic self-refresh, the internal operating clock will be gated for power saving.
	• Supports fast and slow precharge power down in DDR3
	• Automatic active and precharge power down timer per chip select (one chip select can enter power down while the other is still working)

Table continues on the next page...

Table 44-1. MMDC feature summary (continued)

Feature	Details
	• While CS (chip-select) is inactive (high) the command and address buses are not toggling for power saving. • While DM (data masking) is high the associated DQ bus is not toggling (driven to "0") for power saving.

DDR general	• Configurable timing parameters
	• Configurable refresh scheme
	• Page boundary crossing support
	  • Automatically generates precharge command and activates the next row
	• Supports various ODT control schemes
	  • Assertion or deassertion of ODT control per read or write accesses and for active or passive CS (chip-select)
	• Supports MRW and MRR commands for LPDDR2
	• Software control in LPDDR2 mode for switching to derated timing parameters and/or update the refresh rate according to temperature sensor
	• Debug and profiling capabilities

44.2 External Signals

The table found here describes the external signals of MMDC.

Table 44-2. MMDC External Signals

Signal	Description	Pad	Mode	Direction
DRAM_ADDR[15:00]	Address Bus Signals	DRAM_A[15:0]	No Muxing	O
DRAM_CAS	Column Address Strobe Signal	DRAM_CAS	No Muxing	O
DRAM_CS[1:0]	Chip Selects	DRAM_CS[1:0]	No Muxing	O
DRAM_DATA[63:00]	Data Bus Signals	DRAM_D[63:0]	No Muxing	I/O
DRAM_DQM[7:0]	Data Mask Signals	DRAM_DQM[7:0]	No Muxing	O
DRAM_ODT[1:0]	On-Die Termination Signals	DRAM_SDODT[1:0]	No Muxing	O
DRAM_RAS	Row Address Strobe Signal	DRAM_RAS	No Muxing	O
DRAM_RESET	Reset Signal	DRAM_RESET	No Muxing	O
DRAM_SDBA[2:0]	Bank Select Signals	DRAM_SDBA[2:0]	No Muxing	O
DRAM_SDCKE[1:0]	Clock Enable Signals	DRAM_SDCKE[1:0]	No Muxing	O
DRAM_SDCLK[1:0]_N	Negative Clock Signals	DRAM_SDCLK_[1:0]_B	No Muxing	O
DRAM_SDCLK[1:0]_P	Positive Clock Signals	DRAM_SDCLK_[1:0]	No Muxing	O
DRAM_SDQS[7:0]_N	Negative DQS Signals	DRAM_SDQS[7:0]_B	No Muxing	I/O
DRAM_SDQS[7:0]_P	Positive DQS Signals	DRAM_SDQS[7:0]	No Muxing	I/O
DRAM_SDWE	WE signal	DRAM_SDWE	No Muxing	O

44.3 Clocks

The table found here describes the clock sources for MMDC.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 44-3. MMDC Clocks

Clock name	Clock Root	Description
aclk_fast_core_p0	mmdc_ch0_axi_clk_root	Fast clock (channel 1)
aclk_fast_core_p1	GND	Fast clock (channel 2)
ipg_clk_p0	ipg_clk_root	Peripheral clock (channel 1)
ipg_clk_p1	ipg_clk_root	Peripheral clock (channel 2)
aclk_fast_phy_p0	mmdc_ch0_axi_clk_root	Fast clock (channel 1 - PHY)
aclk_fast_phy_p1	GND	Fast clock (channel 2 - PHY)

44.4 Functional Description

This section provides a complete functional description of the block.

44.4.1 Write/Read data flow

44.4.1.1 Write data flow

1. Write requests are received into an 8 entries request FIFO. Access is received only when there are at least two available entries. Each entry holds all of the AXI attributes.
   - If the burst length is greater than 8, the access splits into two accesses: one with burst length 8 and the other with the remainder.
   - The access can be performed as soon as the entire data phase of the associated write request is completed (all data beats were received).
2. A simple round-robin arbitration between the pending read and write accesses is performed, and the pointer to this stage's winner access is sent to the re-ordering buffer.
3. The reordering mechanism is activated to find the winner access, which is the access that best utilizes the DDR bus, based on its dynamic score. For further information see Dynamic scoring mode (Arbitration Winning Conditions).

4. The winner write access at the previous stage is received and is held for dispatch to the DDR logic.
5. When the DDR command control unit is ready to accept the write request, it issues (if needed) a precharge/active command to the DDR device according to the status of the bank model and the parameters of the timers.
6. The DDR logic drives the associated data to the DDR device through the DDR PHY.

44.4.1.2 Read data flow

1. Read requests are received into a 16 entry request FIFO in MMDC if there are at least two available entries. Each entry holds all of the AXI attributes.

NOTE

If the burst length is greater than 8, the access splits into 2 accesses (one with burst length 8 and the other with the remainder).

2. A simple round-robin arbitration between the pending read and write accesses is performed and the pointer to this phase's winner access is sent to the re-ordering buffer.
3. The reordering mechanism is activated to find the winner access, which is the access that best utilizes the DDR bus, based on its dynamic score. For further information see Dynamic scoring mode (Arbitration Winning Conditions).
4. The winner read access at the previous stage is sampled and is held for dispatch to the DDR logic. This read access will be dispatched when there is at least one free slot in the read data buffer to store the data.
5. When the DDR command control unit is ready to accept the read request, it issues (if needed) a precharge/active command to the DDR device according to the status of the bank model and the parameters of the timers.
6. The MMDC PHY samples the read data, and the DDR logic transfers the data to the associated slot in the read data buffer.
7. MMDC transfers the data back to the master.

44.4.2 MMDC initialization

Because the MMDC is disabled when the chip exits reset, no clock is driven to the DDR device and the whole interface towards the DDR device is inactive. The following steps are required to activate the MMDC properly.

NOTE

To guarantee that the DRAM_RESET and DRAM_SDCKE signals are kept low during the power-up and reset sequences of the chip in DDR3 and LPDDR2 modes (as defined by JEDEC), you must connect those signals to pull-down resistors.

1. Set MDSCR[CON_REQ], which sets the configuration request; note that because the MMDC is disabled, there is no need to poll the configuration acknowledge bit at MDSCR[CON_ACK].
2. Configure the desired timing parameters at the MDCFG0, MDCFG1, MDCFG2, and MDOTC registers.
3. Configure the DDR type and other miscellaneous parameters at the MDMISC register.
4. Configure the required delay while leaving reset, at the MDOR register.
5. Configure the DDR physical parameters (density and burst length) at the MDCTL register.
6. Perform a ZQ calibration of the MMDC module to correctly initialize drive strengths.
7. Enable MMDC with the desired chip select at MDCTL[SDE_0] (for chip select 0) and MDCTL[SDE_1] (for chip select 1). At this point, MMDC starts the reset and initialization sequence related to DRAM_RESET/DRAM_SDCKE as defined by JEDEC.
8. Complete the initialization sequence as defined by JEDEC by issuing MRS/MRW commands for (ZQ, ODT, PRE, and so on). To issue those commands, configure the appropriate command and address at the MDSCR register.
9. Program the DDR mode registers by configuring the appropriate command and address at the MDSCR register.
10. Configure the power down and self-refresh entry and exit parameters at the MDPDC and MAPSR registers.
11. Configure the ZQ scheme at the MPZQHWCTRL and MPZQLP2CTRL registers.
12. Configure and activate the periodic refresh scheme at the MDREF register.
13. Deassert the configuration request by clearing MDSCR[CON_REQ].

NOTE

Steps 1 through 5 are non-blocking and can be done in any order.

Upon completion of these steps, MMDC is ready for work and to process AXI accesses.

NOTE

To achieve better timing and better precision, it is recommended that users configure the MMDC PHY delay parameters by operating either the automatic or manual

calibration processs. Before starting any calibration process, you must disable the periodic refresh scheme (MDREF[REF_SEL] = 00) and then issue a manual refresh command by configuring MDSCR[CMD] to 2h. For further information, see Calibration Process.

44.4.3 Configuring the MMDC registers

To safely modify MMDC's internal configuration registers, MMDC must be placed into configuration mode.

Use the following steps to enter configuration mode.

1. Issue a configuration request by setting MDSCR[CON_REQ].
2. Poll on configuration acknowledge until it is set at MDSCR[CON_ACK].

At this point, MMDC enters configuration mode and accessing the MMDC registers is permitted.

NOTE

During configuration mode, MMDC prevents further AXI accesses from being acknowledged.

Upon deassertion of MDSCR[CON_REQ], MMDC leaves configuration mode and AXI accesses are processed.

44.4.4 MMDC Address Space

44.4.4.1 Address decoding

MMDC supports up to two consecutive chip selects, each with the same density. In LPDDR2-2ch mode, up to two chip selects per channel are supported.

It is optional to configure the partition between the chip selects through MDASP[CS0_END].

The incoming AXI address bus is 32 bits. MMDC decodes each access as follows:

1. chip select
2. bank number
3. row number
4. column number

The following registers in the MMDC define the DDR address space:

- MDMISC[DDR_4_BANK]—Defines either 4 or 8 banks in the DDR device
- MDCTL[DSIZ]—Defines the DDR data bus width of x16, x32 or x64
- MDMISC[BI]—Defines whether bank interleaving is on or off
- MDCTL[COL]—Defines the column size of the DDR device
- MDCTL[ROW]—Defines the row size of the DDR device

The following tables show address decoding examples for x16 and x32 bit DDR devices when bank interleaving is both on and off. It is assumed that the configration is as follows: 8 banks (3 bits), 15 bit assignment for the row, and 10 bit sassignment for the column. The total density is 256 MWords (512 Mbytes for x16 and 1 Gbyte for x32).

NOTE

Chip selection is done by comparing the 7 most significant address bits (ARADDR[31:25]/AWADDR[31:25]) with MDASP[CS0_END].

Table 44-4. Address decoding—bank interleaving off

AXI ADDRESS	x16 DDR	x32 DDR
A29	—	BANK[2]
A28	BANK[2]	BANK[1]
A27	BANK[1]	BANK[0]
A26	BANK[0]	ROW[14]
A25	ROW[14]	ROW[13]
A24	ROW[13]	ROW[12]
A23	ROW[12]	ROW[11]
A22	ROW[11]	ROW[10]
A21	ROW[10]	ROW[9]
A20	ROW[9]	ROW[8]
A19	ROW[8]	ROW[7]
A18	ROW[7]	ROW[6]
A17	ROW[6]	ROW[5]
A16	ROW[5]	ROW[4]
A15	ROW[4]	ROW[3]
A14	ROW[3]	ROW[2]
A13	ROW[2]	ROW[1]
A12	ROW[1]	ROW[0]
A11	ROW[0]	COL[9]
A10	COL[9]	COL[8]
A9	COL[8]	COL[7]
A8	COL[7]	COL[6]
A7	COL[6]	COL[5]

Table continues on the next page...

Table 44-4. Address decoding—bank interleaving off (continued)

AXI ADDRESS	x16 DDR	x32 DDR
A6	COL[5]	COL[4]
A5	COL[4]	COL[3]
A4	COL[3]	COL[2]
A3	COL[2]	COL[1]
A2	COL[1]	COL[0]
A1	COL[0]	—
A0	—	—

Table 44-5. Address decoding—bank interleaving on

AXI ADDRESS	x16 DDR	x32 DDR
A29	—	ROW[14]
A28	ROW[14]	ROW[13]
A27	ROW[13]	ROW[12]
A26	ROW[12]	ROW[11]
A25	ROW[11]	ROW[10]
A24	ROW[10]	ROW[9]
A23	ROW[9]	ROW[8]
A22	ROW[8]	ROW[7]
A21	ROW[7]	ROW[6]
A20	ROW[6]	ROW[5]
A19	ROW[5]	ROW[4]
A18	ROW[4]	ROW[3]
A17	ROW[3]	ROW[2]
A16	ROW[2]	ROW[1]
A15	ROW[1]	ROW[0]
A14	ROW[0]	BANK[2]
A13	BANK[2]	BANK[1]
A12	BANK[1]	BANK[0]
A11	BANK[0]	COL[9]
A10	COL[9]	COL[8]
A9	COL[8]	COL[7]
A8	COL[7]	COL[6]
A7	COL[6]	COL[5]
A6	COL[5]	COL[4]
A5	COL[4]	COL[3]
A4	COL[3]	COL[2]
A3	COL[2]	COL[1]
A2	COL[1]	COL[0]

Table continues on the next page...

Table 44-5. Address decoding—bank interleaving on (continued)

AXI ADDRESS	x16 DDR	x32 DDR
A1	COL[0]	—
A0	—	—

NOTE

In cases where this is an access to a non-initialized or disconnected chip select, behavior may be unexpected.

44.4.4.2 Chip select settings

MMDC drives the incoming access to either CS0 or CS1 by comparing the 7 most significant address bits (ARADDR[31:25]/AWADDR[31:25]) with MDASP[CS0_END].

Generally, the total density per chip-select must be the same, and the total density per chip-select must be a power of two.

Creating 4 Gbyte address space with 2 Gbyte CS density and Creating 2 Gbyte address spaces with 1 Gbyte CS density show how to create a continous address space and configure the MMDC accordingly.

44.4.4.2.1 Creating 4 Gbyte address space with 2 Gbyte CS density

If the DDR memory space allocation is 4 Gbytes, only one configuration of chip select partition is allowed.

The register MDASP[CS0_END] should be set to 011_1111 (partition at 2 Gbytes).

The figure below shows the associated memory space. In the case of DDR3 x64, this address space can be acheived by connecting four devices per chip select. Each device is x16 with density of 4 Gbytes.

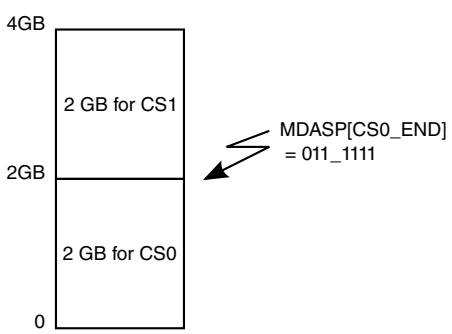


Figure 44-2. Chip select partion—2 Gbytes per chip select

44.4.4.2.2 Creating 2 Gbyte address spaces with 1 Gbyte CS density

If the DDR memory space allocation is 2 Gbytes, there are three options for configuring the chip select partition:MDASP[CS0_END] to 001_1111 (1 Gbyte), MDASP[CS0_END] to 011_1111 (2 Gbytes), and MDASP[CS0_END] to 101_1111 (3 Gbytes).

If DDR memory space allocation is 2 Gbytes, there are three options for configuring the chip select partition:

- MDASP[CS0_END] to 001_1111 (1 Gbyte)
- MDASP[CS0_END] to 011_1111 (2 Gbytes)
- MDASP[CS0_END] to 101_1111 (3 Gbytes)

The figure below shows the associated memory space:

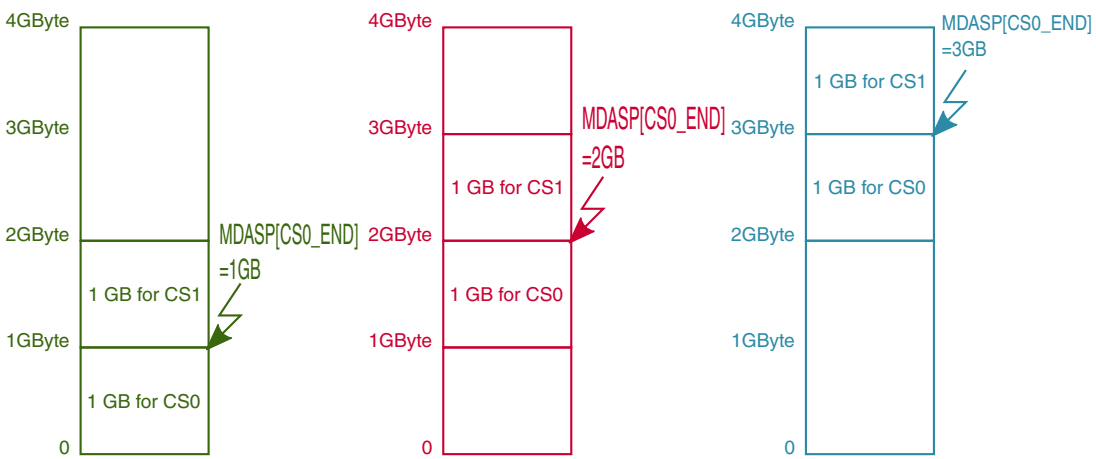


Figure 44-3. Chip select partion—1 Gbyte per chip select

44.4.4.3 Translation of AXI accesses to DDR accessess

44.4.4.3.1 Example 1

Assume the AXI read access has the following attributes:

- Wrap (arburst[1:0] = 10b)
- AXI size of 128 bits (arsize[2:0] = 100b)
- AXI length of 8 (arlen[3:0] = 0111b)
- AXI address with suffix B0h (non-aligned to AXI wrap boundary which is 16Bx8 = 128B = 0x80)

Toward DDR3(MDMISC[DDR_TYPE] = 00b) with the following attributes:

- x32 (MDCTL[DSIZ] = 01b)
- burst length of 8 (MDCTL[BL] = 1b)

In this case, the AXI wrap boundary is every 16Bx8=128B (0x80) and the DDR wrap boundary is every 4Bx8=32B(0x20).

The master expects to fetch the data that is associated with the following addresses: 0xB0, 0xC0, 0xD0, 0xE0, 0xF0, 0x80, 0x90, 0xA0.

The first aligned AXI address that is associated with suffix 0xB0 is with suffix 0x80, and the last wrap AXI address is with suffux 0xFF. Because the AXI master expects to get the first data from AXI address with suffix 0xB0, the MMDC issues the following accesses toward the DDR:

- Read access toward logic address with suffix 0xA0 (DDR boundary is 0x20 and 0xA0 is the closest to 0xB0)

NOTE

Logic address is the address of the column normalized to 1 byte.

- Read access toward logic address with suffix 0xC0
- Read access toward logic address with suffix 0xE0
- Read access toward logic address with suffix 0x80

The MMDC breaks the AXI access into four DDR accesses and returns the read data associated with address 0xB0 to the master first. The read data fetched from address 0xA0 is stored in the internal buffers and is driven back to the master at the end.

44.4.4.3.2 Example 2

Assume the AXI write access has the following attributes:

- Increment (awburst[1:0]=2'b01)
- AXI size of 64bits (awsize[2:0]=3'b011)
- AXI length of 8 (awlen[3:0]=4'b0111)
- AXI address with suffix 0xB0 (aligned as the size of the increment is 8B)

Toward DDR3(MDMISC[DDR_TYPE]=2'b00) with the following attributes:

- x64 (MDCTL[DSIZ]=2'b10)
- burst length of 8 (MDCTL[BL]=1'b1)

In this case, the AXI alignment is every 8B and the DDR boundary is every 8Bx8=64B(0x40).

The master expects to write the data to the following addresses: 0xB0, 0xB8, 0xC0, 0xC8, 0xD0, 0xD8, 0xE0, 0xE8.

The MMDC will issue the following accesses toward the DDR:

- Write access toward logic address with suffix 0x80 (DDR boundary is 0x40 and 0x80 is the closest to 0xB0) while address 0x80 to 0xAF are masked by DM (data masking signal).

NOTE

Logic address is the address of the column normalized to 1 byte.

- Write access toward logic address with suffix 0xC0 while addresses 0xF0 through 0xFF are masked by DM (data masking signal)

The MMDC will break the AXI access into two DDR accesses.

44.4.4.3.3 Example 3

Assume the AXI write access has the following attributes:

- Wrap (awburst[1:0]=2'b10)
- AXI size of 128bits (awsize[2:0]=3'b100)
- AXI length of 4 (awlen[3:0]=4'b0011)
- AXI address with suffix 0x80 (aligned)

Toward DDR3(MDMISC[DDR_TYPE]=2'b00) with the following attributes:

- x64 (MDCTL[DSIZ]=2'b10)
- burst length of 8 (MDCTL[BL]=1'b1)

In this case, the AXI wrap boundary is every 16Bx4=64B (0x40) and the DDR wrap boundary is every 8x8=64B(0x40).

The master expects to write the data to the following addresses: 0x80, 0x90, 0xA0, 0xB0.

Because the AXI wrap boundary and DDR wrap boundary are similar and the starting AXI address is aligned, the MMDC will issue only one access toward the DDR as follows:

- Write access towards logic address with suffix 0x80

NOTE

Logic address is the address of the column normalized to 1 byte.

44.4.4.3.4 Example 4

Assume the AXI write access has the following attributes:

- Increment (awburst[1:0]=2'b01)
- AXI size of 64bits (awsize[2:0]=3'b011)
- AXI length of 2 (awlen[3:0]=4'b0001)
- AXI address with suffix 0x5 (non aligned)

Toward DDR3(MDMISC[DDR_TYPE]=2'b00) with the following attributes:

- x32 (MDCTL[DSIZ]=2'b10)
- burst length of 8 (MDCTL[BL]=1'b1)

In this case the AXI alignment is every 8B (0x8) and the DDR boundary is every 4Bx8=32B(0x20).

The master expects to write the data to the following addresses: 0x5 (with WSTRB=0xE0), 0x8 (till 0xF).

The MMDC will issue one access toward the DDR as follows:

Write access toward logic address with suffix 0x0 (DDR boundary is 0x20 and 0x0 is the closest to 0x0) while address 0x0 till 0x4 are masked by DM (data masking signal) and address 0x10 till 0x1F are also masked by DM.

44.4.4.3.5 Example 5

Assume AXI write access has the following attributes:

- Increment (awburst[1:0]=2'b01)
- AXI size of 64bits (awsize[2:0]=3'b011)
- AXI length of 7 (awlen[3:0]=4'b0001)
- AXI address with suffix 0x10 (aligned)

Toward DDR3 (MDMISC[DDR_TYPE]=2'b00) with the following attributes:

- x64 (MDCTL[DSIZ]=2'b10)
- burst length of 8 (MDCTL[BL]=1'b1)

In this case the AXI alignment is every 8B (0x8) and the DDR boundary is every 8Bx8=64B(0x40).

The master expects to write the data to the following addresses: 0x10, 0x18, 0x20, 0x28, 0x30, 0x38, 0x40, 0x48.

Because the AXI access is not aligned to DDR boundary, which is every 0x40, the MMDC will issue two accesses toward the DDR as follows:

- Write access toward logic address with suffix 0x0 while address 0x0 till 0xF are masked by DM (data masking signal).

NOTE

Logic address is the address of the column normalized to 1 byte.

- Write access towards logic address with suffix 0x40 while addresses 0x50 till 0x7F are masked by DM (data masking signal).

44.4.4.4 Address mirroring

When enabling this feature, address bits DRAM_A3, DRAM_A4, DRAM_A5, DRAM_A6, DRAM_A7, DRAM_A8, DRAM_SDBA0, and DRAM_SDBA1 behave differently according to the associated chip select.

This feature facilitates PCB board routing for devices on chip select 1, which are typically populated on the opposite side of the PCB from the devices on chip select 0.

NOTE

This feature is only supported for DDR3 memories. It is not supported for LPDDR2 memories.

The following table specifies the address mirroring options:

Table 44-6. Address mirroring options

MMDC pin	Chip select 0 pin	Chip select 1 pin
DRAM_A3	DRAM_A3	DRAM_A4
DRAM_A4	DRAM_A4	DRAM_A3
DRAM_A5	DRAM_A5	DRAM_A6
DRAM_A6	DRAM_A6	DRAM_A5
DRAM_A7	DRAM_A7	DRAM_A8
DRAM_A8	DRAM_A8	DRAM_A7
DRAM_SDBA0	DRAM_SDBA0	DRAM_SDBA1
DRAM_SDBA1	DRAM_SDBA1	DRAM_SDBA0

44.4.5 LPDDR2 and DDR3 pin mux mapping

The following table shows the pin mux mapping between LPDDR2 and DDR3. The i.MX DDR I/O pads corresponds with the DDR3 standard.

- In DDR3, all DRAM_DATA, DRAM_SDQS, and DRAM_DQM data lines work with channel 0.
- In LPDDR2, DRAM_DDQS[3:0], DRAM_DATA[31:0] and DRAM_DQM[3:0] work with channel 0. DRAM_SDQS[7:4], DRAM_DATA[63:32], and DRAM_DQM[7:4] work with channel 1.

Table 44-7. LPDDR2 and DRAM pin mux mapping

DRAM I/O pad	LPDDR2 functionality
DRAM_ADDR00	LPDDR2_CA9_P0
DRAM_ADDR01	LPDDR2_CA7_P0
DRAM_ADDR02	LPDDR2_CA8_P0
DRAM_ADDR03	LPDDR2_CA1_P1
DRAM_ADDR04	LPDDR2_CA0_P1
DRAM_ADDR05	LPDDR2_CA3_P1
DRAM_ADDR06	LPDDR2_CA2_P1
DRAM_ADDR07	LPDDR2_CKE0_P1
DRAM-ADDR08	LPDDR2_CA4_P1
DRAM_ADDR09	LPDDR2_CKE1_P1
DRAM_ADDR10	LPDDR2_CA6_P0
DRAM_ADDR11	LPDDR2_CS_B1_P1
DRAM_ADDR12	LPDDR2_CS_B0_P1
DRAM_ADDR13	LPDDR2_CA1_P0
DRAM_ADDR14	LPDDR2_CA5_P1
DRAM_ADDR15	LPDDR2_CA7_P1
DRAM_CAS_B	LPDDR2_CA5_P0
DRAM_RAS_B	LPDDR2_CS_B1_P0
DRAM_WE_B	LPDDR2_CKE0_P0
DRAM_SDCKE0	LPDDR2_CA9_P1
DRAM_CKE1	LPDDR2_CA8_P1
DRAM_CS_B0	LPDDR2_CKE1_P0
DRAM_CS_B1	LPDDR2_CA2_P0
DRAM_ODT0	LPDDR2_CA3_P0
DRAM_ODT1	LPDDR2_CA0_P0
DRAM_SDCLK0_P	LPDDR2_CK_P0
DRAM_SDCLK1	LPDDR2_CK_P1
DRAM_BA0	LPDDR2_CS_B0_P0
DRAM_BA1	LPDDR2_CA4_P0
ddr3_ba2	LPDDR2_CA6_P1

44.4.6 Power Saving and Clock Frequency Change modes

44.4.6.1 Power saving general

MMDC supports multiple DDR power saving modes.

NOTE

At default, the power saving modes are disabled. These modes may dramatically decrease the power consumption of DDR memories.

1. Self-refresh entry to the entire DDR device (for both chip select 0 and 1) can be activated through two mechanisms:
   - LPMD (Low Power Mode)
     - Hardware handshaking (LPMD/LPACK) with the clock module in the system
     - Software handshaking by setting the field MAPSR[LPMD] and polling MAPSR[LPACK]
     - Automatic entry by configuring the amount of idle cycle for triggering self-refresh entry through MAPSR[PST] and by clearing MAPSR[PSD]
   - DVFS (Dynamic Voltage and Frequency Change)
     - Hardware handshaking (DVFS/DVACK) with the clock module in the system
     - Software handshaking by setting the field MAPSR[DVFS] and polling MAPSR[DVACK]

NOTE

If hardware or software requests for self-refresh entry were detected by the MMDC (even before the assertion of the LPACK), no write or read accesses will be acknowledged utnil the deassertion of those requests.

2. Automatic active/precharge power down entry to a specific chip select can be activated by configuring the ESDPDC register:
   - PWDT_0/PWDT_1 - define the number of idle cycles before entering power down, can be different value per chip select.
   - SLOW_PD - In case of DDR3 memory is configured to use slow precharge power down then this bit should be set as well.
   - BOTH_CS_PS - The MMDC can either set each chip select independently to power down, according to its idle state, or set both chip selects to power down only if both in idle state for the configured period.
   - Few paramters must be configured in addition:

- Timing parameters at ESDCFG0[tXP and tXPDLL].
- ODT timing at ESDOTC[tAOFPD, tAONPD, tANPD and tAXPD]

NOTE

It is possible to enter certain chip selects to low power consumption while the second chip select is activated.

3. Automatic precharge of all DDR banks to a specific chip select. Can be activated by configuring ESDPDC fields: PRCT_0 and PRCT_1. Each field determines a value loaded to a different chip select.

44.4.6.2 Self refresh and Frequency change entry/exit

As described in Power saving general, the MMDC supports two mechanisms that will cause the DDR device to enter self-refresh mode:

- LPMD (Low Power Mode) - For power saving purposes
- DVFS (Dynamic Voltage and Frequency Change) - For clock frequency changes

While the DDR device is in self-refresh mode, there is no need to provide periodic refresh commands.

The MMDC treates hardware/software handshaking of LPMD/DVFS in the same manner:

- Upon the assertion of LPMD/DVFS request, the following is done:
  - The MMDC blocks any further AXI accesses even before the acknowledge is asserted
  - Completes all opened AXI accesses
  - Closes (precharge) all banks in the appropriate timing
  - Drives self-refresh command by deasserting clock enable signal (DRAM_SDCKE is driven to "0") together with a refresh command. This occurs after satisfying tRP/tRPA from the precharge all command.
  - Deasserts the clock (CK) that is driven to the DDR device
  - Asserts LPMD/DVFS acknowledge (LPACK/DVACK)
  - Allows deassertion of the operating clock of the MMDC (AXI clock)
- Upon the deassertion of LPMD/DVFS request, the following is done:
  - Operating clock of the MMDC must be turned on before LPMD/DVFS is deasserted
  - Starts driving the clock (CK) to the DDR device
  - After satisfying tCKSRX from clock renewal the clock enable signal (DRAM_SDCKE) is asserted
  - LPMD/DVFS acknowledge (LPACK/DVACK) is deasserted

- After satisfying tXS from the assertion of DRAM_SDCKE, a refresh command is driven to the DDR device.
- If ZQ calibration is enabled then tRFC is satisfied from the refresh command and a long ZQ command is driven.
- tZQoper idle cycles are counted after the ZQ command.
- After satisfying tDLLK from the assertion DRAM_SDCKE, the MMDC returns to normal operation.

The figure below shows the timing diagram of the hardware/software handshaking of LPMD/DVFS:

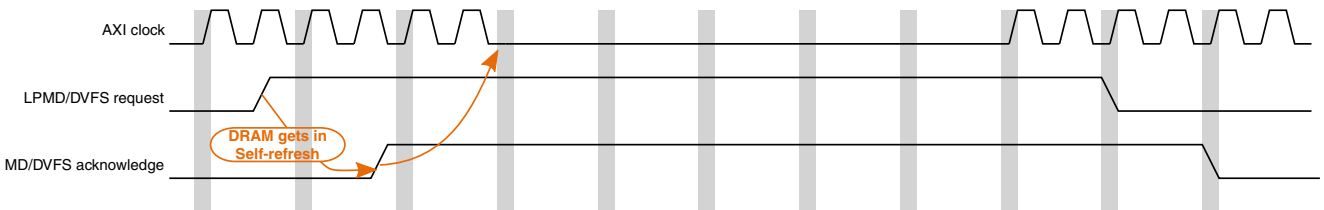


Figure 44-4. LPMD/DVFS Hardware/Software Handshaking

Note for self-refresh:

- As soon as LPMD or DVFS requests are detected by either hardware or software handshaking, the MMDC will deassert the AXI ARREADY/AWREADY signals immediately to block further requests from the system.
- In case of automatic self-refresh, the internal operating clock will be negated to save power.

44.4.7 Reset

44.4.7.1 Hard reset

When hard reset is asserted (aresetn is driven to "0") while warm reset is deasserted (warm_reset is driven to "0"), the entire MMDC will be initialized, including configuration/status registers and state machines.

In order to access the DDR device, the MMDC will then have to be reconfigured.

44.4.7.2 Warm reset

The MMDC supports warm reset signal. The warm reset signal must envelop the hard reset signal and then the MMDC will reset all the internal registers. The only registers that are not reset are those that are essential for returning it to normal operation without repeating the initialization sequence and without losing data stored in the memory (configuration/status registers won't be initialized).

For the successful operation of warm reset, the following steps must be performed:

- The MMDC must enter self-refresh mode. This can be achieved by either LPMD or DFVS requests
- Wait for LPMD or DVFS acknowledge
- Assert warm reset signal (i.e. drive warm_reset to "1")
- Assert hard reset signal (i.e. drive aresetn to "0")
- Deassert hard reset signal
- Deassert warm reset
- Get out of the LPMD/DVFS mode

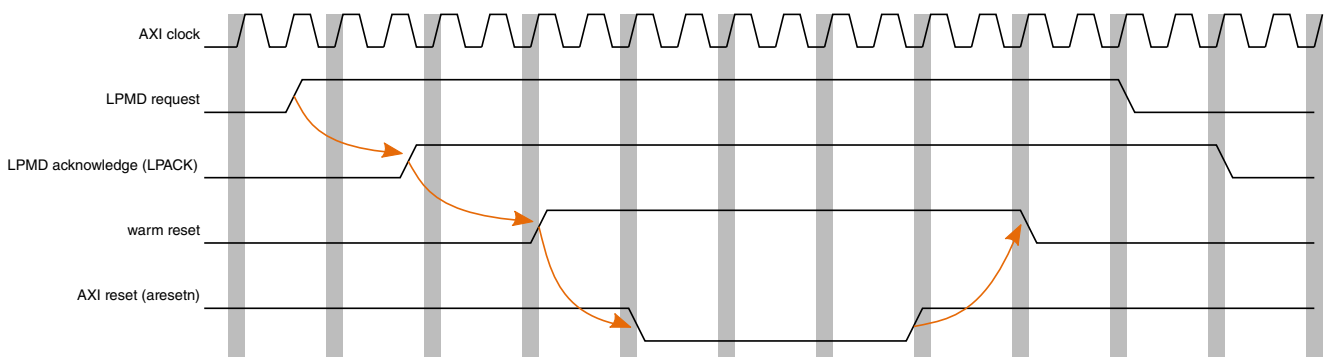


Figure 44-5. Warm Reset Diagram

44.4.7.3 Software reset

The MMDC supports software reset. When software reset is configured then the MMDC will reset all the internal registers except those that are essential for returning to normal operation without repeating the initialization sequence or without losing data stored in the memory (configuration/status registers won't be initialized).

The following steps should be performed for successful operation of software reset:

- The MMDC should enter self-refresh mode. This can be achieved by either LPMD or DFVS request.
- Wait for LPMD or DVFS acknowledge

- Assert software reset, by setting MDMISC[RST]
- Get out of the LPMD/DVFS mode

Normal operation can be resumed.

44.4.8 Refresh Scheme

The MMDC supports various automatic refresh options which can be configured via the MDREF register.

The periodic auto refresh can be triggered by the following clocks:

- 32KHz clock
- 64KHz clock
- MMDC operating clock

The refresh scheme of the MMDC is flexible and allows the system to configure the desired AXI accesses delay/latency in each refresh cycle.

The table below shows an example of four configurations of the refresh cycles that will be handled by the MMDC. Each configuration meets a refresh rate of 3.9us (tREFI, refresh command every 3.9us).

Table 44-8. MMDC Refresh Scheme

Option number	Description	REFR	REF_SEL	REF_CNT	DDR hang time
1	Issue 8 refresh commands every 31,250 ns	0x7 (8 refreshes)	0x2 (64KHz)	not needed	tRFC * 8
2	Issue 4 refresh commands every 15,625ns	0x3 (4 refreshes)	0x1(32KHz)	not needed	tRFC * 4
3	Issue 2 refresh commands every 7800ns	0x1(2 refreshes)	0x3 (fast counter)	7800/2.5 = 3120 (0xC30)	tRFC * 2
4	Issue 1 refresh command every 3900 ns	0x0 (1 refresh)	0x3 (fast counter)	3900/2.5 = 1560(0x618)	tRFC

44.4.9 Burst Length options towards DDR

The MMDC supports two kinds of burst lengths which can be configured through MDCTL[BL] as follows:

- In DDR3 mode, only burst length 8 can be used.
- In LPDDR2 mode, only burst length 4 can be used.

In DDR3 mode read/write accesses to the DDR are always 8 words (x16, x32, x64) and aligned in according to JEDEC standards.

In case of AXI INCREMENT, accesses that are not aligned the irrelevant data is masked in write acesses and ignored in read accesses. In case of AXI WRAP accesses, even if the access is not aligned, then the MMDC provides an internal optimization mechanism for better efficiency of the DDR data bus.

44.4.10 Exclusive accesses handling

The MMDC contains four exclusive monitors, each for dedicated ID as configured in MAEXIDR0 and MAEXIDR1.

- If legal read exclusive is received by the MMDC, the associated monitor is turned on.
- While the monitor is turned on upon legal write exclusive, the monitor will be turned off and the write will be completed successfully with EXOKAY.
- The following rules must be met for successful exclusive access:
  - Aligned access (the AXI address is aligned to the AXI size)
  - AXI single access (AXI burst length isn't greater than 1)
  - AXI size of up to 64 bits
  - AXI non-cachable access (i.e. ARCACHE[1]/AWCACHE[1] is equal "0" or ARCACHE[1]/AWCACHE[1] is equal "1" while ARCACHE[3:2]/ AWCACHE[3:2] are equal "00")
  - AXI ID that matches one of the four exclusive IDs

Exclusive read behavior (first bullet also correct for non-exclusive accesses):

- In case of security violation, the read is blocked and is not sent to DDR. There are two options for response:
  - If ARCR_SEC_ERR_EN (MAARCR[30]) is high, SLV error is issued towards the Master, otherwise OKAY response is sent to the Master.
- If AXI exclusive rules violation occurs (as described above), the read access is not blocked and is sent to DDR. The data will be fetched and be driven to the master, but the type of response may be unpredicted.
- If none of the above occurs, the read is sent to the DDR. The exclusive monitor will be turned on and the response is ExOKAY
- If additional legal AXI read exclusive is received with the same ID before the AXI exclusive write, the monitor will be updated with the latest attributes.

Exclusive write behavior (first bullet also correct for non-exclusive accesses):

- In case of security violation, the write is blocked and is not sent to DDR, but the monitor will be kept on. There are two options for response:
  - If ARCR_SEC_ERR_EN (MAARCR[30]) is high then SLV error is issued towards the Master, otherwise OKAY response is sent to the Master.
- In case of AXI exclusive rules violation (as described above), the write is blocked and is not sent to DDR. In that case the type of response may be unpredicted.
- In case the exclusive write access has different AXI attributes, but the same ID as the read exclusive access, the write is blocked and is not sent to DDR and the monitor will be turned off. There are two options for response:
  - If ARCR_EXC_ERR_EN (MAARCR[28]) is high then SLV error is issued towards the Master, otherwise OKAY response is sent to the Master.
- In case of regular (non exclusive) write access is received to the same address or overlapping addresses then the write will be sent to the DDR and the monitor will be turned off.
- In case of legal write exclusive access is received with the same attributes as the read exclusive access while the monitor is on (no write accesses occured to the same address between the read exclusive and write exclusive), then the write is sent to DDR and the response is EXOKAY. But, if the legal write exclusive is received while the monitor is off, the write is blocked and there are two options for response.
  - If ARCR_EXC_ERR_EN (MAARCR[28]) is high then SLV error is issued towards the Master, otherwise OKAY response is sent to the Master.

44.4.11 AXI Error Handling

The MMDC supports the AXI responses listed here.

- In case of AXI exclusive violation there are two options for response:
  - If MAARCR[28] is high then SLVError is issued towards the Master, Otherwise OKAY response is sent to the Master

NOTE

In case of read error MMDC drives zeros on the read data bus

44.5 Performance

44.5.1 Arbitration and reordering mechanism

44.5.1.1 Arbitration General

The following specifies arbitration and reordering flow in MMDC towards the DDR.

- AXI read and write accesses are sampled in the associated queue.
- Read/write arbitration is handled to select the winning access.
- Winning access is sampled in the reordering queue
- Reordering mechanism is handled between valid requests that reside in the reordering queue to select the access that will be dispatched to the DDR.
    - The reordering is held in order to optimize the accesses and to maximize the utilization of the DDR bus
    - As soon as the reordered access is completed (indicated by end of response or data phase) then it is erased from the associated queue and the MMDC is ready to receive the next available access from the master

In general, the reordering/arbitration mechanism is based on dynamic priority mechanism, which compares dynamic priorities between valid entries in the reordering queue and issues the entry with highest dynamic priority towards the DDR Logic.

The selection of the winning access is based on two modes, which can be activated together, as following:

- Real time channel mode:
    - Accesses with QoS='f' (i.e. awqos[3:0]/arqos[3:0] = "f") will bypass all other requests towards the DDR
- Dynamic scoring mode:
    - The arbitration mechanism is based on dynamic priority. Relevant for the accesses with QoS smaller than 'f' or when real time channel mode is disabled.

NOTE

Due to re-ordering and optimization mechanism (per different AXI ID), the transactions towards the DDR may be driven in a different ID order they were received by the AXI master. In similar way, the write response, read response or read data may be driven to the AXI master in a different ID order.

44.5.1.2 Real time channel mode

When real time mode is enabled (i.e MAARCR[ARCR_RCH_EN] = "1") , all requests with QoS='f' (i.e. awqos[3:0]/aqqos[3:0] = "f") will bypass all other pending accesses towards the DDR. This mode is enabled by default.

44.5.1.3 Dynamic scoring mode (Arbitration Winning Conditions)

The arbitration between pending accesses in the MMDC is handled according to a dynamic priority of each access.

The dynamic priority (may be also called score) is calculated according to a sum of some factors (final_score[3:0]), where part of them may be updated dynamically. The following will specify each scoring factor:

- MAARCR[ARCR_PAG_HIT] (Page hit score) - A static score which is taken into account in case the pending access has a page hit
- MAARCR[ARCR_ACC_HIT] (Access hit score) - A static score, which is taken into account in case the current access type (read/write) is the same as the access that has been dispatched to the DDR previously
- MAARCR[ARCR_DYN_JMP] (Dynamic jump score) - A dynamic score which is given to any pending access in case it was not chosen in the arbitration. The dynamic jump counter is limited by maximum value which is set in MAARCR[ARCR_DYN_MAX] .
- QoS score which is indicated through a sideband 4bits AXI signals (awqos[3:0]/ aqqos[3:0]) and is driven by the AXI master per access

Note: In order to prevent an overflow in the total sum of scores, a clipping is held and selects the maximum score value of 'f' once a total scores sum is greater than 'f'.

The figure below shows the dynamic score calculations

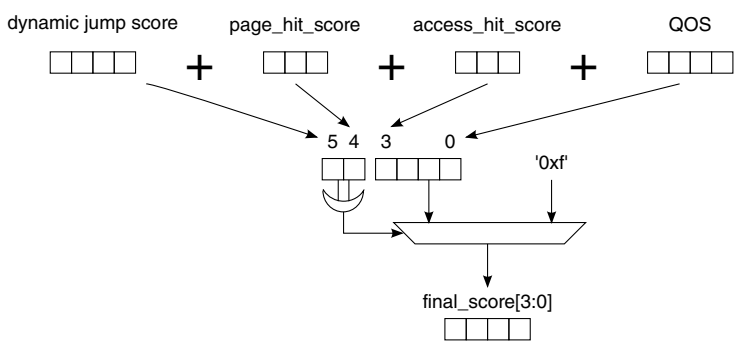


Figure 44-6. Dynamic score/priority calculation

44.5.1.4 Guarding (aging) mechanism

The guarding mechanism (may be also called aging) is used to prevent a starvation of accesses.

As soon as the dynamic jump score reaches its maximum value (MAARCR[ARCR_DYN_MAX]) then each time a pending request was not chosen in the arbitration, the "guarding" counter is incremented by 1. When the "guarding" counter reaches its predefined value, set in MAARCR[ARCR_GUARD], the associated request gets the highest priority and will be chosen in the next arbitration cycle towards the DDR unless a real time channel (i.e access with QoS ="f") is arrived.

Note: In case real time channel has arrived then the dynamic score of the non real time channels won't increment in order to prevent a case where the "guarding" counter of more than one access has reached its limit.

44.5.2 Prediction mechanism

When prediction mechanism is enabled (i.e by configuring MDMISC[MIF3_MODE]) then the MMDC predicts the chip-select, bank address and row address that is going to be issued towards the DDR before the access is physically dispatched towards DDR device.

That mechanism enables to prepare the DDR device with future accesses and improves the overall DDR performance.

This prediction mechanism operates in parallel to the reordering mechanism and may yield a prediction based on 3 levels of pending accesses:

1. Access in first stage of pipeline.
2. Valid access on AXI bus either read channel or write channel.
3. Valid access on special bus from arbitration - this access is chosen by the arbitration as the next miss access in its buffers

44.5.3 Special Optimization for accesses towards DDR3

In case an AXI read/write wrap non-aligned access is acknowledged in DDR3 mode with the same wrap boundary as the DDR wrap boundary then the MMDC will make an optimization and issue only one access towards the DDR, although all the accesses towards the DDR3 must be aligned.

For example: AXI write access with size of 128bits (awsize[2:0]=3'b100), length of 4 (awlen[3:0]=4'b0011) towards DDR3 x64 (burst length 8). In that case the AXI wrap boundary is 16Bx4=64B (0x40) and the DDR3 wrap boundary is 8Bx8=64B (0x40). If, for example, the AXI access is towards AXI address with suffix of 0x10 (non-aligned to 64B boundary) then the MMDC will get from the AXI master the data that is associated with addresses 0x10, 0x20, 0x30, 0x0. The MMDC will rearrange internally the data so it

will match DDR3 alignment as following: 0x0, 0x10, 0x20, 0x30 and drive it in one access towards the DDR to address 0x0. The alternative was to issue two accesses towards the DDR with address 0x0 with different data masking

NOTE

In read wrap access the same optimization is handled, while as soon as the critical AXI word is fetched from the DDR then it is driven immediately to the AXI master without buffering. Based on the example above, the master expects to fetch first the data that is associated with address 0x10. Therefore the MMDC will issue read access from address 0x0 of the DDR and as soon as the data that is associated with address 0x10 is received then it will be driven back immediately to the master even before fetching the data of the furhter addresses.

44.6 MMDC Debug

44.6.1 Hardware debug monitor

The MMDC has a hardware debugging mechanism that monitors each access that is driven to the MMDC from channel 0.

Every time this mechanism is enabled (setting of MADPCR0[DBG_EN] to "1") then each access that will be dispatched to the DDR will be also observed in the I/O pads (i.e. over ipp_do_ddr_debug[50:0]). The content of this bus is described in the table below.

Table 44-9. Hardware monitor debugging

Signal Name	Number of Bits	Description
acc_addr	[31:0]	AXI ADDRESS of the selected access
acc_type	1	access type of the selected access. "0 indicates write. "1" indicates read.
acc_id	[15:0]	AXI transaction ID of the selected access
valid_strobe	1	indication for a valid request . This signal will be asserted for 1 clock cycle

The fields above are organized as following:

MMDC_DEBUG[50:0] = {1'b0,valid_strobe,acc_id,access_type,addr}

These signals are sent to IOMUX, in IOMUX user can configure it to be output from the chip for debug usage.

44.6.2 Step By Step (SBS) software monitor

The MMDC has a Step By Step (SBS) software debugging mechanism that monitors each access that is driven to the MMDC.

Every time this mechanism is trigerred then one AXI access will be dispatched to the DDR and in parallel its attributes will be observed in a status register.

Once the "step by step" is enabled (i.e. MADPCR0[SBS_EN] is "1") then all accesses to the DDR device will be halted.

Setting MADPCR0[SBS] to "1" will dispatch the access that is pending in the head of the MMDC queue (read or write). Upon every setting of MADPCR0[SBS]:

- The AXI attributes of the access will be sampled in the associated MASBS0 and MASBS1 fields
- MADPCR0[SBS] will be cleared automatically.

Setting again MADPCR0[SBS] to "1" will dispatch the next pending access in the MMDC queue.

44.7 MMDC Profiling

The profiling mechanism provides the ability to calculate the DDR utilization together with read and write accesses statistics towards DDR per given period of time.

MMDC supports the following profiling counters:

- MADPSR0 (Total cycles count) - Indicates the total amount of cycles of the profiling period (up to 2^32 cycles)
- MADPSR1 (Busy cycles count) - Indicates the total busy cycles during the profiling period. Busy cycles are any MMDC clock cycles where the internal state machine is not idle. If any read or write requests are pending in the FIFOs, the MMDC is not idle.
- MADPSR2 (Total read accesses count) - Indicates the total read accesses towards MMDC during the profiling period
- MADPSR3 (Total write accesses count) - Indicates the total write accesses towards MMDC during the profiling period
- MADPSR4 (Total read bytes count) - Indicates total bytes that were read from MMDC during the profiling period
- MADPSR5 (Total write bytes count) - Indicates total bytes that were written to MMDC during the profiling period

All profiling items described above are disabled by default. The following describes how to control the profiling mechanism:

- MADPCR0[DBG_EN] enables profiling.
- MADPCR0[PRF_FRZ] stops/freezes the profiling for example in case user wishes to perform DDR profiling per specific task. In order to resume profiling then MADPCR0[PRF_FRZ] should be cleared.
- MADPCR0[DBG_RST] clears all profiling counters
- MADPCR0[CYC_OVF] indicates whether an overflow occured in the total cycles counter (i.e. total amount of cycles are greater than 2^32) . This field can only cleared by writing '0'.

Read/Write statistics can be collected per specific AXI ID (16bits). The following fields in MADPCR1 register determines which AXI-ID or AXI-ID's to monitor:

- PRF_AXI_ID defines which AXI IDs are taken for profiling. Default values is 16'h0.
- PRF_AXI_ID_MASK defines which bits from PRF_AXI_ID will be compared with AXI ID of read/write access. "1" means to monitor the associated bit and "0" means don't care. Default value is 16'h0000, meaning all IDs are not monitored

So the AXI-IDs to be monitored are calculated according to the following equation:

(AXI-ID & PRF_AXI_ID_MASK) Xnor (PRF_AXI_ID & PRF_AXI_ID_MASK)

For example if AXI ID's between A100 till A1FF are wished to be monitored then the following should be configured:

- PRF_AXI_ID = A100
- PRF_AXI_ID_MASK = FF00

Table 44-10. i.MX 6Dual/6Quad AXI ID

Master	AXI ID at DDR controllers ('x' denotes ID from master)
ARM_S0	14'b000xxxxxxxx000
ARM_S1	14'b000xxxxxxxx001
IPU1	14'b000000000xx100
IPU2	14'b000000000xx101
GPU3D_a	14'b0000xxxx000010
GPU2D_a	14'b0000xxxx001010
VDOA	14'b000000xx010010
OpenVG	14'b0000xxxx100010
HDMI	14'b00000100011010
SDMA (Burst)	14'b00000101011010
SDMA (Periph)	14'b00000110011010
CAAM	14'bxxxx0000011010

Table continues on the next page...

Table 44-10. i.MX 6Dual/6Quad AXI ID (continued)

Master	AXI ID at DDR controllers ('x' denotes ID from master)
USB	14'b00xx0001011010
ENET	14'b00000010011010
HSI	14'b00000011011010
uSDHC1	14'b00000111011010
GPU3D_b	14'b0000xxxx000011
GPU3D_b	14'b0000xxxx001011
VPU Prime	14'b0000xxxx010011
PCIe	14'b000xxxxx011011
DAP	14'b00000000100011
APBH DMA	14'b00000010100011
BCH40	14'bxxxx0001100011
SATA	14'b00000011100011
MLB150	14'b00000100100011
uSDHC2	14'b00000101100011
uSDHC3	14'b00000110100011
uSDHC4	14'b00000111100011

44.8 LPDDR2 Refresh Rate Update and Timing Derating

LPDDR2 devices may have a temperature sensor that is used to determine an appropriate refresh rate and whether AC timing derating is required. The status of the temprature sensor can be read through MRR command from LPDDR2 MR4 register.

The MMDC supports refresh update and timing derating mechanism on the fly. The following specify how to use that mechanism:

- Perform periodic polling on MR4 LPDDR2 register using MRR command
- Read MDMRR register and analyze the MR4 indication
- In case refresh rate update and/or AC timing derating is required then it is needed to update MDREF and/or MDMR4[tRCD_DE, tRC_DE, tRAS_DE, tRP_DE, tRRD_DE] parameters

NOTE

MDMR4[tRCD_DE, tRC_DE, tRAS_DE, tRP_DE, tRRD_DE] are referred to the associated values configured at MDCFG3LP[tRC_LP, tRP_LP, tRCD_LP], MDCFG1[tRAS], MDCFG2[tRRD]

- Assert MDMR4[UPDATE_DE_REQ]
- When the MMDC switch to the new values then an acknowledge will be indicated at MDMR4[UPDATE_DE_ACK]

44.9 DLL Off mode

DLL Off mode is supported only in DDR3 and allows operation of the DDR in low frequency (i.e. below 125MHz as defined in JEDEC standard).

For further details refer to DLL-off Mode chapter in the standard.

The following steps should be executed in order to switch from DLL on to DLL off mode:

- Assert CON_REQ signal and wait to CON_ACK assertion.
- Disable power down timers that can conflict with this sequence, such as: MAPSR[PSD], MDPDC[PWDT_1], MDPDC[PWDT_0], MDPDC[PRCT_0], MDPDC[PRCT_1].
- Execute precharge all banks command (via MDSCR).
- Execute MRW command to MR1 and disable RTT Nom (A9,A6,A2 =0) and DLL ON (A0 =1).
- Execute MRW command to MR2 in order to update CWL to 6.
- Execute MRW command to MR0 in order to update CL to 6.
- De-assert CON_REQ signal.
- Enter self refresh mode. For further information refer to Self refresh and Frequency change entry/exit.
- At self refresh entry acknowledge , change to the desired frequency.
- Exit self refresh mode.
- Assert CON_REQ and wait to CON_ACK assertion.
- Enable Pull Down resistors on DQS (through the I/O-MUX).
- Configure the MMDC register as following:
- Update tCWL =6 and tCL =6 to meet the values configured in the DDR device. (MDCGFG0, MDCFG1)
- Disable ODT resistor (i.e. set MPODTCTRL to "0").
- Disable DQS gating (i.e. set MPDGCTRL0[DG_DIS] to "1").
- Enable required power down timers that were disabled, such as: MAPSR[PSD], MDPDC[PWDT_1], MDPDC[PWDT_0], MDPDC[PRCT_0], MDPDC[PRCT_1].
- De-assert CON_REQ and wait for de-assertion of CON_ACK.

The following steps should be executed in order to switch from DLL off to DLL on:

- Execute precharge all banks command (via MDSCR).

- Enter self refresh mode. For further information refer to Self refresh and Frequency change entry/exit.
- At self refresh entry acknowledge, change to the desired frequency.
- Exit self refresh mode
- Assert CON_REQ and wait to CON_ACK assertion.
- Disable power down timers that can conflict with this sequence, such as: MAPSR[PSD], MDPDC[PWDT_1], MDPDC[PWDT_0], MDPDC[PRCT_0], MDPDC[PRCT_1].
- Execute MRW command to MR1 and enable RTT Nom (A9,A6,A2) and DLL ON (A0 =0).
- Execute MRW command to MR0 to reset the DLL (A8) and update CL value
- Execute MRW command to MR2 in order to update CWL value.
- Execute ZQ commnad.
- Reconfigure MMDC BLOCK
- Update tCWL and tCL to meet the values configured to the memory. (MDCGFG0, MDCFG1)
- Enable ODT resistor (i.e. MPODTCTRL register)
- Enable DQS gating (i.e. set MPDGCTRL0[DG_DIS] to "0").
- Disable Pull Down resistors on DQS (through the I/O-MUX).
- Enable required power down timers that were disabled, such as: MAPSR[PSD], MDPDC[PWDT_1], MDPDC[PWDT_0], MDPDC[PRCT_0], MDPDC[PRCT_1].
- De-assert CON_REQ and wait for de-assertion of CON_ACK.

44.10 ODT Configuration

The MMDC supports two DRAM_ODT signals (DRAM_ODT for each DRAM_CS) in DDR3 mode in order to allow the DDR device to turn on/off its termination resistors. The MMDC suggests various configuration for the assertion of the ODT signals as well as configuration of several related timing.

The following specifies the options for configuring the assertion of DRAM_ODT signals:

- Assert DRAM_ODT signal for the non active DRAM_CS in write. For example : when this bit asserted - if writing to DRAM_CS0 the DRAM_ODT of DRAM_CS1 will be asserted. This is done by setting MPODTCTRL[0] to "1"
- Assert DRAM_ODT signal for the active DRAM_CS in write by by setting MPODTCTRL[1] to "1"
- Assert DRAM_ODT signal for the non active DRAM_CS in read by by setting MPODTCTRL[2] to "1"
- Assert DRAM_ODT signal for the active DRAM_CS in read by by setting MPODTCTRL[3] to "1"

MDOTC register controls the timing for the DRAM_ODT signals assertion.

NOTE

tODTLon determines the delay between DRAM_ODT signal and the associated RTT, where according to JEDEC standard it equals WL(write latency) - 2. Therefore, the value configured to MDOTC[tODTLon] field should correspond with the value configured to MDCGFG1[tCWL].

In precharge power down mode , when all banks are closed, the assertion of ODT corresponds with tAOFPD and tAONPD which are configured in MDOTC register.

The figure below shows timing diagram of DRAM_ODT and RTT signals while MPODTCTRL[0] is set to "1" (i.e. assertion of DRAM_ODT to the non active DRAM_CS in write access command) and MDOTC[tODTLon] is set to 4.

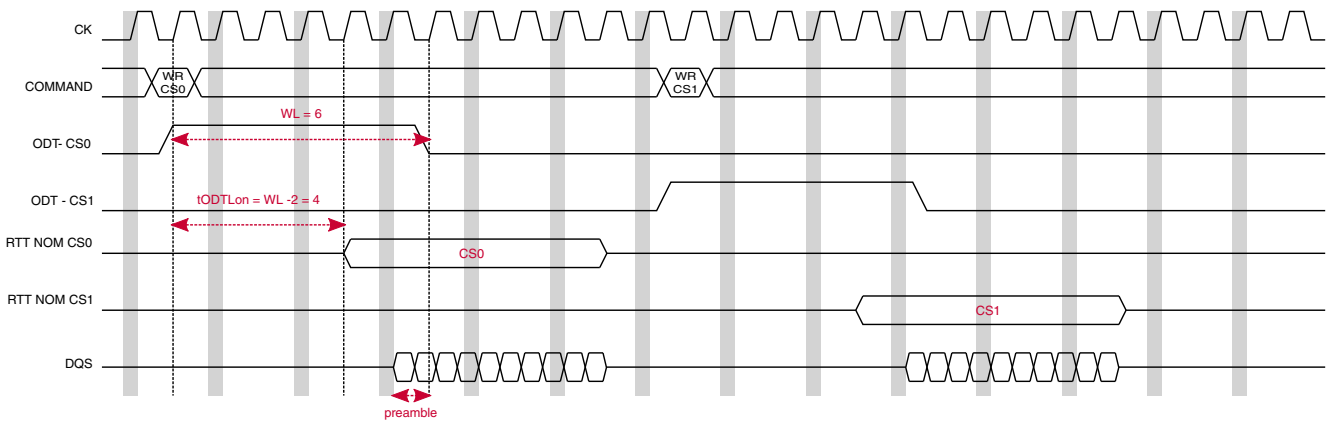


Figure 44-7. ODT - Timing Diagram DDR3 WL=6, BL=8

44.11 Calibration Process

The MMDC offers various calibration processes that are used to obtain better timing accuracy, board skew compensation and I/O pad driving strength adjustment.

Each calibration process can be performed either automatically (hardware) or manually (software), though the manual method is typically reserved for debugging purposes. The following calibration processes are supported:

NOTE

Power saving features should be disabled before the calibration process begin. (Such as: MDPDC[PWDT#], MDPDC[PRCT#], MAPSR[PSD])

- ZQ calibration for external DDR device (in DDR3 through ZQ calibration command and in LPDDR2 through MRW command)
  - Can be handled automatically for ZQ Short (periodically) and ZQ Long (at exit from self-refresh)
  - Can be handled manually at ZQ INIT
- ZQ calibration for i.MX DDR I/O pads for calibrating the DDR driving strength
  - The sequence can be handled automatically by hardware
  - The sequence can be handled step by step manually by software
- Read DQS gating calibration for DDR3 only. Adjustment of DQS gate with read preamble window. For further information refer to Read DQS Gating Calibration
- Read data calibration. Adjustment of read DQS with read data byte. For further information refer to Read Calibration
- Write data calibration. Adjustment of write DQS with write data byte. For further information refer to Write Calibration
- Write leveling calibration. Adjustment of write DQS with CK (DDR differential clock). For further information refer to Write leveling Calibration
- Read fine tuning. Adjustment of up to 7 delay-line units for each read data bit.
- Write fine tuning. Adjustment of up to 3 delay-line units for each read data bit.

NOTE

Before starting any calibration process that involves the DDR3 device MPR mode or write leveling calibration, the following should be done:

- Disable the periodic refresh scheme (i.e. setting MDREF[REF_SEL] = "00") and then issue manual refresh command burst by configuring MDSCR[CMD]= 0x2. At the end of the calibration it is needed to enable the periodic refresh scheme.
- Disable the automatic power saving mode (i.e set MAPSR[PSD] = "1").

44.11.1 Delay-line

Each of the calibration processes controls several delay-lines for aligning data and strobes.

By default the delay-line is configured to generate 1/4 clock cycle of delay. The maximum delay that may be issued by the delay-line, while configured to the value 127, is as following:

- Under best-case conditions, -40C, 1.21V - 1.6ns.
- Under worst-case conditions, 125C, 0.99V - 3.8ns

Moreover, when the operating clock is at the maximum allowed frequency, as appeared in the features list, then the delay-line is capable to issue a configurable delay of up to 1/2 clock cycle.

NOTE

At the beginning of the calibration process the initial value of the delay-line must be a valid value (i.e. the strobes must be somewhere among the associated data window) though it might not be the optimal value. The delay-line calibration should be done after Read DQS gating and write-leveling calibrations.

In order to generate an adequate delay during normal operation of the MMDC the delay-line is going through an automatic measurement process during the refresh period of the DDR device

44.11.2 ZQ calibration

The MMDC supports ZQ calibration process to calibrate the driving strength of the i.MX DDR I/O pads as well as driving ZQ commands to calibrate the external DDR device driving strength.

The first i.MX ZQ calibration (after booting the processor) is performed prior to turning on the MMDC. Subsequent i.MX ZQ calibrations may be executed in parallel to the DDR ZQ calibration. The MMDC supports 2 types of ZQ calibration commands: short and long.

The ZQ long calibration is executed during power up sequence, when existing self-refresh mode or when exiting slow precharge power down (DLL lock can be done in parallel). The ZQ short calibration is executed periodically according to a configurable timer defined by MPZQHWCTRL[ZQ_HW_PER].

The field MPZQHWCTRL[ZQ_MODE] determines whether the MMDC will execute ZQ calibration to i.MX DDR I/O pads and/or issue ZQ short/long command to the DDR device.

The MMDC supports both automatic (hardware) and manual (software) ZQ calibration process for the i.MX DDR I/O pads.

It is possible to perform automatic (hardware) ZQ calibration only once (i.e. non-periodical) by asserting MPZQHWCTRL[ZQ_HW_FOR].

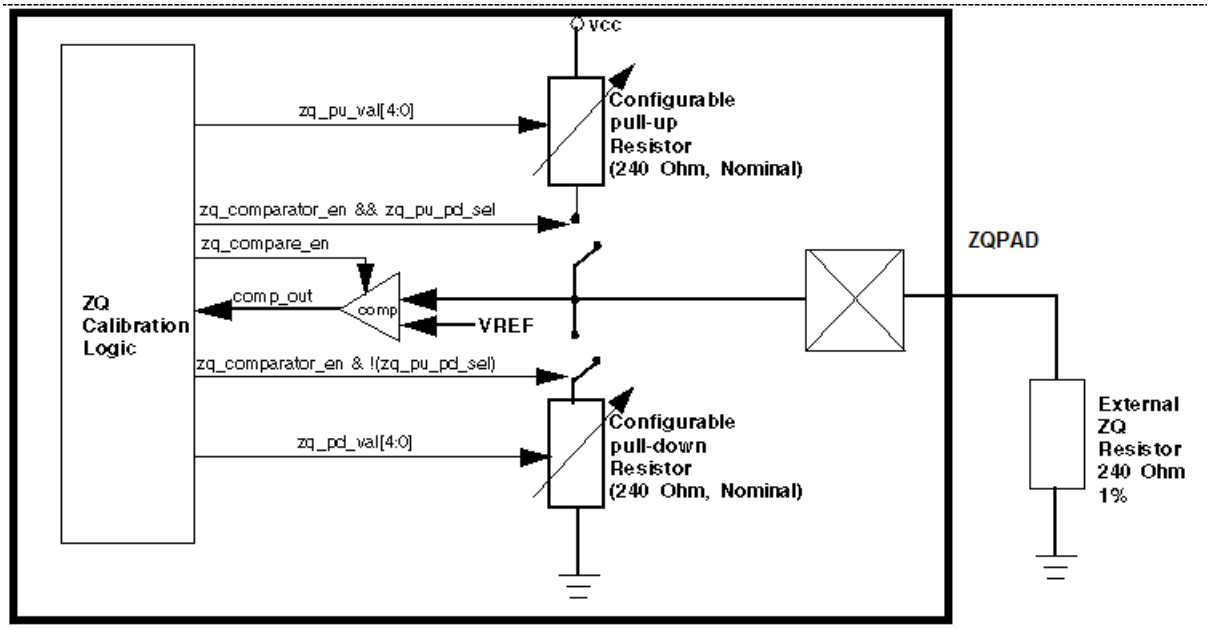


Figure 44-8. MMDC ZQ IF with PAD

44.11.2.1 ZQ automatic (hardware) calibration process

The ZQ automatic calibration lasts 11 steps. 5 steps for the pull up resistors calibration and 6 for the pull down resistors calibration.

The calibration control interface with ZQ pin is described below:

The calibration process is as follows:

44.11.2.1.1 ZQ automatic Pull-up calibration

The MMDC automatically performs a handshaking mechanism with the i.MX ZQ calibration pad as follows:

1. The MMDC drives zq_comparator_en to "1"
2. The MMDC waits few cycles according to MPZQHWCTRL[EARLY_COMPARATOR_EN_TIMER]
3. The MMDC drives zq_pu_pd_sel to "1" for indication of pull-up calibration and drives zq_pu_val[4:0] = 5'b00000
4. MMDC drives zq_pu_val[4] to "1"
5. MMDC asserts zq_compare_en
6. MMDC waits few cycles according to MPZQSWCTRL[ZQ_CMP_OUT_SMP] before sampling the comparator output (i.e zq_comp_out). If zq_comp_out is "1" then it means that the output voltage is greater than Vdd/2 (i.e. internal resistor is less than 240 ohm) and drives bit zq_pu_val[4] to "1" else it drives zq_pu_val[4] to "0"

7. MMDC deasserts zq_compare_en
8. MMDC repeats steps 4- 7for zq_pu_val bits 3 to 0
9. MMDC drives ZQ calibration result to MPZQHWCTRL[ZQ_HW_PU_RES]
10. MMDC advances to pull-down calibration

44.11.2.1.2 ZQ automatic Pull-down calibration

1. The MMDC drives zq_pu_pd_sel to "0" for indication of pull-down calibration and drives zq_pd_val[4:0] = 5'b00000
2. MMDC drives zq_pd_val[4] to "1"
3. MMDC asserts zq_compare_en
4. MMDC waits few cycles according to MPZQSWCTRL[ZQ_CMP_OUT_SMP] before sampling the comparator output (i.e zq_comp_out). If zq_comp_out is "1" then it means that the output voltage is greater than Vdd/2 (i.e. internal resistor is less than 240 ohm) and drives bit zq_pd_val[4] to "0" else it drives zq_pd_val[4] to "1"
5. MMDC deasserts zq_compare_en
6. MMDC repeats steps 12- 15 for zq_pd_val bits 3 to 0
7. MMDC drives ZQ calibration result to MPZQHWCTRL[ZQ_HW_PD_RES]
8. MMDC deassert zq_comparator_en to indicate the completion of the ZQ calibration

44.11.2.2 ZQ software calibration process

The ZQ calibration can be done also in software. However since software ZQ calibration is much slower than hardware calibration it should be used mainly for debugging.

Software should configure the ZQ calibration parameters (Pull-up or Pull-down and their value) then assert the MPZQSWCTRL[ZQ_SW_FOR] bit. Then software should wait till ZQ_SW_FOR is de-asserted and use ZQ_SW_RES status bit in order to calculate the next ZQ calibration parameters.

44.11.2.3 ZQ calibration commands

Before the MMDC can issue a ZQCL/ZQCS command to the memory it should precharge all memory banks and wait tRP period. A single ZQ command can be issued to all devices as long as the devices don't share the same ZQ resistor.

When the MMDC issues the ZQ command it should also drive A10 (long or short command) and CS (0, 1 or both).

The MMDC must keep the memory lines quiet (except for CK) for the ZQ calibration time as defined in the Jedec (512 cycles for ZQCL after reset, 256 for other ZQCL and 64 for ZQCS).

44.11.3 Read DQS Gating Calibration

The read DQS gating calibration is used to adjust the read DQS gating with the middle of the read DQS preamble.

The DQS gating includes a delay of up to 7 cycles (The delay is chosen according to two fields MPDGCTRL#[DG_HC_DEL#] and MPDGCTRL#[DG_DL_ABS_OFFSET#]

Each DQS has its own delay-line. The DQS gating process can be done for all DQS in parallel.

NOTE

In LPDDR2 mode hardware Read DQS gating should be disabled and Pull-up/pull-down resistors on DQS/DQS# should be enabled while ODT resistors must be disconnected.

In DDR3_x64 mode activation of the calibration is done by setting MPDGCTRL0[HW_DG_EN] at address 0xBASE0_083C

44.11.3.1 Hardware DQS Gating Calibration

- There are two modes of operations:
  - Calibration with the MPR (Multi Purpose Register)
  - Calibration with MMDC pre-defined values

44.11.3.1.1 Hardware DQS Calibration with MPR

The following steps should be executed:

1. Precharge all active banks (Can be done through MDSCR) as requried by the standard.
2. Enter the DDR device into MPR mode through MRS commands
3. Configure the MMDC to work with MPR mode by asserting MPPDCMPR2[MPR_CMP]
4. Make sure that the initial value that is configured in the read delay line absolute offset of each byte (i.e. MPRDDLCTL[RD_DL_ABS_OFFSET#]) will place the read DQS somewhere inside the read DQ window
5. Start the calibration process by asserting MPDGCTRL0[HW_DG_EN]

44.11.3.1.2 Hardware DQS Calibration with pre-defined value

In case pre-defined mode is used, (i.e. MPPDCMPR2[MPR_CMP]) is cleared, then the following steps should be executed:

1. Precharge all active banks (Can be done through MDSCR) as required by the standard.
2. Configure the pre-defined value, which reflects the value that will be written and compared through the read calibration, to MPPDCMPR1[PDV1, PDV2]
3. Issue write access to the external DDR device by setting MPSWDAR0[SW_DUMMY_WR] = 1 (MMDC will generate internally write access without intervention of the system towards bank 0, row 0, column 0)
4. Make sure that the initial value that is configured in the read delay line absolute offset of each byte (i.e. MPRDDLCTL[RD_DL_ABS_OFFSET#]) will place the read DQS somewhere inside the read DQ window
5. Start the calibration process by asserting MPDGCTRL0[HW_DG_EN]

   The following steps will be executed automatically by the MMDC for both modes (MPR and Pre-defined value):

6. MMDC waits till the read DQS delay-line is updated with the absolute delay value for all bytes at MPDGCTRL#[DG_HC_DEL#] and MPDGCTRL#[DG_DL_ABS_OFFSET#] and also satisfying the Tmod + 4 requirement
7. MMDC drives read command to the external DDR devices and waits 16 or 32 cycles (according to MPDGCTRL0[DG_CMP_CYC] assuming that the data has arrived from the DDR device.
8. MMDC compares the read data byte to the associated byte in the pre-defined/MPR value for all the bytes in the DDR burst (burst length 4 or 8). If the comparison fails then it indicates that the read DQS gating is asserted in illegal time point. If the comparison passes then MMDC advances to step 14
9. MMDC resets the read FIFO (to the inverted pre-defined/MPR value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1
10. MMDC increments the read DQS gating delay of each byte by half cycle (i.e. MPDGCTRL#[DG_HC_DEL#] + 1)
11. MMDC drives read command to the external DDR devices and waits 16 or 32 cycles (according to MPDGCTRL0[DG_CMP_CYC] assuming that the data has arrived from the DDR device.
12. MMDC compares the read data byte to the associated byte in the pre-defined/MPR value for all the bytes in the DDR burst (burst length 4 or 8).
13. If the comparison fails then it indicates that the read DQS gating is asserted in illegal time point and it is needed to repeat steps 9-12. If the comparison passes then MMDC stores the value of the temporary low boundary and advances to next step

14. MMDC increments the read DQS gating delay-line of each byte by half cycle (i.e. MPDGCTRL#[DG_HC_DEL#] + 1) and issue measurement process of the read DQS gating delay-line to update itself with the new value
15. MMDC drives read command to the external DDR devices and waits 16 or 32 cycles (according to MPDGCTRL0[DG_CMP_CYC] assuming that the data has arrived from the DDR device.
16. MMDC compares the read data byte to the associated byte in the pre-defined/MPR value for all the bytes in the DDR burst (burst length 4 or 8)
17. If the comparison passes then it indicates that the read DQS gating is asserted inside the read preamble window and it is needed to repeat steps 14-16. If the comparison fails then MMDC stores the value of the temporary upper boundary and starts searching the adequate low and high boundaries
18. MMDC returns to the temporary low boundary minus half cycle and issue measurement process of the read DQS gating delay-line to update itself with the new value
19. MMDC drives read command to the external DDR devices and waits 16 or 32 cycles (according to MPDGCTRL0[DG_CMP_CYC] assuming that the data has arrived from the DDR device.
20. MMDC compares the read data byte to the associated byte in the pre-defined/MPR value for all the bytes in the DDR burst (burst length 4 or 8)
21. If the comparison fails then it indicates that the read DQS gating is asserted in illegal time point and it is needed to repeat steps 22-23. If the comparison passes then MMDC stores the value of the adequate low boundary and advances to step 24
22. MMDC resets the read FIFO (to the inverted pre-defined/MPR value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1
23. MMDC increments the read DQS gating delay of each byte by 1 (i.e. MPDGCTRL#[DG_DL_ABS_OFFSET#] + 1) and issue measurement process of the read DQS gating delay-line to update itself with the new value and advances to step 19
24. MMDC returns to the temporary upper boundary minus half cycle and issue measurement process of the read DQS gating delay-line to update itself with the new value
25. MMDC drives read command to the external DDR devices and waits 16 or 32 cycles (according to MPDGCTRL0[DG_CMP_CYC] assuming that the data has arrived from the DDR device.
26. MMDC compares the read data byte to the associated byte in the pre-defined/MPR value for all the bytes in the DDR burst (burst length 4 or 8)
27. If the comparison passes then it indicates that the read DQS gating is asserted inside the read preamble window and it is needed to repeat steps 28-29. If the comparison fails then MMDC stores the value minus 1 of the adequate upper boundary and advances to step 30

28. MMDC resets the read FIFO (to the inverted pre-defined/MPR value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1
29. MMDC increments the read DQS gating delay of each byte by 1 (i.e. MPDGCTRL#[DG_DL_ABS_OFFSET#] + 1) and issue measurement process of the read DQS gating delay-line to update itself with the new value and advances to step 25
30. After the MMDC finds the window boundary (lower and upper) of each read data byte then it stores the average between lower and upper boundaries at the associated MPDGCTRL#[DG_DL_ABS_OFFSET#] and issue measurement process of the read DQS delay-line to update itself with the new value.
31. MMDC indicates that the read DQS gating calibration had finished by setting MPDGCTRL0[HW_DG_EN] = 0
32. Exit the DDR device from MPR mode through MRS command
33. Read the upper boundary that was found: MPDGHWST#[HW_DG_UP#]. This field is 11 bits, 7 LSB bits correspond to MPDGCTRL#[DG_DL_ABS_OFFSET#] upper limit value and 4 MSB bits correspond to MPDGCTRL#[DG_HC_DEL#] upper limit value.
34. Set MPDGHWST#[HW_DG_UP#][6:0] to MPDGCTRL#[DG_DL_ABS_OFFSET#].
35. Set (MPDGHWST#[HW_DG_UP#][10:7] - 1) to MPDGCTRL#[DG_HC_DEL#]. (We set the DQS gating value to be the upper limit value minus 1 half cycle)

44.11.3.2 SW read DQS gating Calibration

- There are two modes of operations:
- Calibration with the MPR (Multi Purpose Register)
- Calibration with MMDC pre-defined values

44.11.3.2.1 SW read Calibration with MPR

The following steps should be executed:

1. Precharge all active banks (Can be done through MDSCR) as required by the standard.
2. Enter the DDR device into MPR mode through MRS commands
3. Configure the MMDC to work with MPR mode by asserting MPPDCMPR2[MPR_CMP]
4. Make sure that the initial value that is configured in the read delay line absolute offset of each byte (i.e. MPRDDLCTL[RD_DL_ABS_OFFSET#]) will place the read DQS somewhere inside the read DQ window

44.11.3.2.2 SW read Calibration with pre-defined value

In case pre-defined mode is used, (i.e. MPPDCMPR2[MPR_CMP]) is cleared, then the following steps should be executed:

1. Precharge all active banks (Can be done through MDSCR) as required by the standard.
2. Configure the pre-defined value, which reflects the value that will be written and compared through the read calicration, to MPPDCMPR1[PDV1, PDV2]
3. Issue write access (with any legal DDR address) to external DDR device
4. Make sure that the initial value that is configured in the read delay line absolute offset of each byte (i.e. MPRDDLCTL[RD_DL_ABS_OFFSET#]) will place the read DQS somewhere inside the read DQ window

   The following steps should be executed automatically by the MMDC for both modes (MPR and Pre-defined value):

5. Configure the read DQS delay-line to issue zero delay by setting MPDGCTRL#[DG_DL_ABS_OFFSET#] = 0 and MPDGCTRL#[DG_HC_DEL#] = 0
6. Force the delay line to measure itself and to issue the requested read delay by configuring MPMUR[FRC_MSR] = 1
7. Wait 16 DDR cycles till the read DQS delay-line is updated with the absolute delay value for all bytes
8. Issue read command (with the legal DDR address chosen in step 3) from the external DDR device
9. Waits 16 or 32 cycles (according to MPDGCTRL0[DG_CMP_CYC]) assuming that the data has arrived from the DDR device.
10. Compare the read data byte to the associated byte in the pre-defined/MPR value for all the bytes in the DDR burst (burst length 4 or 8). If the comparison fails then it indicates that the read DQS gating is asserted in illegal time point. If the comparison passes then advance to step 15
11. MMDC resets the read FIFO (to the inverted pre-defined/MPR value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1
12. Increment the read DQS gating delay of each byte by half cycle (i.e. MPDGCTRL#[DG_HC_DEL#] + 1)
13. MMDC drives read command to the external DDR devices and waits 16 or 32 cycles (according to MPDGCTRL0[DG_CMP_CYC]) assuming that the data has arrived from the DDR device.
14. Compare the read data byte to the associated byte in the pre-defined/MPR value for all the bytes in the DDR burst (burst length 4 or 8). If the comparison fails then it indicates that the read DQS gating is asserted in illegal time point and it is needed to repeat steps 11 - 14. If the comparison passes then advance to step 15

15. Store the temporary lower boundary and start searching the temporary upper boundary
16. MMDC resets the read FIFO (to the inverted pre-defined/MPR value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1
17. Increment the read DQS gating delay of each byte by half cycle (i.e. MPDGCTRL#[DG_HC_DEL#] + 1)
18. MMDC drives read command to the external DDR devices and waits 16 or 32 cycles (according to MPDGCTRL0[DG_CMP_CYC] assuming that the data has arrived from the DDR device.
19. Compare the read data byte to the associated byte in the pre-defined/MPR value for all the bytes in the DDR burst (burst length 4 or 8).
20. If the comparison passes then it indicates that the read DQS gating is asserted inside the read preamble window and it is needed to repeat steps 16-19. If the comparison fails then it is needed to store the value of the temporary upper boundary and starts searching the adequate low and high boundaries
21. Load the temporary low boundary minus half cycle into the associated MPDGCTRL#[DG_HC_DEL#]
22. Reset the read FIFO (to the inverted pre-defined/MPR value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1
23. Increment the read DQS gating delay of each byte by 1 (i.e. MPDGCTRL#[DG_DL_ABS_OFFSET#] + 1) and force the delay line to measure itself and to issue the requested read DQS delay by configuring MPMUR[FRC_MSR] = 1
24. Issue read command to the external DDR devices and waits 16 or 32 cycles (according to MPDGCTRL0[DG_CMP_CYC]) assuming that the data has arrived from the DDR device.
25. Compare the read data byte to the associated byte in the pre-defined/MPR value for all the bytes in the DDR burst (burst length 4 or 8)
26. If the comparison fails then it indicates that the read DQS gating is asserted in illegal time point and it is needed to repeat steps 22-26. If the comparisons passes then advance to the next step.
27. Store the adequate lower boundary
28. Load the temporary upper boundary minus half cycle into the associated MPDGCTRL#[DG_HC_DEL#]
29. Reset the read FIFO (to the inverted pre-defined/MPR value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1
30. Increment the read DQS gating delay of each byte by 1 (i.e. MPDGCTRL#[DG_DL_ABS_OFFSET#] + 1) and force the delay line to measure itself and to issue the requested read DQS delay by configuring MPMUR[FRC_MSR] = 1

31. Issue read command to the external DDR devices and waits 16 or 32 cycles (according to MPDGCTRL0[DG_CMP_CYC] assuming that the data has arrived from the DDR device.
32. Compare the read data byte to the associated byte in the pre-defined/MPR value for all the bytes in the DDR burst (burst length 4 or 8)
33. If the comparison passes then it is needed to repeat steps 29-32. If the comparisons fails then advance to the next step.
34. Reset the read FIFO (to the inverted pre-defined/MPR value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1
35. Store the adequate upper boundary.
36. Keep the MPDGCTRL#[DG_DL_ABS_OFFSET#] value of the upper limit.
37. Set MPDGCTRL#[DG_HC_DEL#] = (MPDGCTRL#[DG_HC_DEL#] - 1). (We set the DQS gating value to be the upper limit value minus 1 half cycle)
38. Issue the requested read DQS delay by configuring MPMUR[FRC_MSR] = 1
39. Exit the DDR device from MPR mode through MRS command

44.11.4 Read Calibration

The read calibration is used to adjust the read DQS with read data byte.

It is assumed that the read DQS gating calibration process is completed prior to the read calibration.

NOTE

In DDR3 mode, the activation of the calibration is done by setting MPRDDLHWCTL[HW_RD_DL_EN] at address 0xBASE0_0860 In LP2_x16, LP2_x32 the activation of the calibration of each channel is done by setting MPRDDLHWCTL[HW_RD_DL_EN] at address 0xBASE0_0860 and MPRDDLHWCTL[HW_RD_DL_EN] at address 0xBASE1_0860 respectively.

44.11.4.1 Hardware (automatic) Read Calibration

- There are two modes of operations:
- Calibration with the MPR (Multi Purpose Register)/DQ calibration(LPDDR2)
- Calibration with MMDC pre-defined values

44.11.4.1.1 Hardware (automatic) Calibration with MPR (DDR3) /DQ Calibration (LPDDR2)

The following steps should be executed:

1. Precharge all active banks (can be done through MDSCR) as requried by the standard.
2. Enter the DDR device into MPR/DQ calibration mode through MRS/MRW commands.
3. Configure the MMDC to work with MPR/DQ calibration mode by asserting MPPDCMPR2[MPR_CMP].
4. Make sure that the initial value that is configured in the read delay line absolute offset of each byte (MPRDDLCTL[RD_DL_ABS_OFFSET#]) will place the read DQS somewhere inside the read DQ window.
5. Start the calibration process by asserting MPRDDLHWCTL[HW_RD_DL_EN].

44.11.4.1.2 Hardware (automatic) Calibration with pre-defined value

In case pre-defined mode is used, i.e. MPPDCMPR2[MPR_CMP] is cleared, then the following steps should be executed:

1. Precharge all active banks (Can be done through MDSCR) as required by the standard.
2. Configure the pre-defined value, which reflects the value that will be written and compared through the read calibration, to MPPDCMPR1[PDV1, PDV2]
3. Issue write access to the external DDR device by setting MPSWDAR0[SW_DUMMY_WR] = 1 (MMDC will generate internally write access without intervention of the system towards bank 0, row 0, column 0)
4. Make sure that the initial value that is configured in the read delay line absolute offset of each byte (i.e. MPRDDLCTL[RD_DL_ABS_OFFSET#]) will place the read DQS somewhere inside the read DQ window
5. Start the calibration process by asserting MPRDDLHWCTL[HW_RD_DL_EN]

   The following steps will be executed automatically by the MMDC for both modes (MPR and Pre-defined value):

6. MMDC waits till the read delay-line is updated with the absolute delay value for all bytes at MPRDDLCTL[RD_DL_ABS_OFFSET#] and also satisfying the Tmod + 4 requirement
7. MMDC drives read command to the external DDR devices and waits 16 or 32 cycles (according to MPRDDLHWCTL[HW_RD_DL_CMP_CYC]) assuming that the data has arrived from the DDR device.
8. MMDC compares the read data byte to the associated byte in the pre-defined/MPR value for all the bytes in the DDR burst (burst length 4 or 8). If the comparison fails

then it indicates that the initial read DQS isn't inside the read DQ window and the MMDC generates an error for the associated byte at MPRDDLHWCTL[HW_RD_DL_ERR#] . If the comparison passes then MMDC advances to next step.

9. MMDC resets the rd fifo (to the inverted pre-defined/MPR value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1
10. MMDC decrements the read delay line absolute offset of each byte by 1 (i.e. MPRDDLCTL[RD_DL_ABS_OFFSET#]) and issue measurement process of the read delay-line to update itself with the new value.
11. MMDC drives read command to the DDR external devices and waits 16 or 32 cycles (according to MPRDDLHWCTL[HW_RD_DL_CMP_CYC]) assuming that the data has arrived from the DDR device
12. MMDC compares the read data byte to the associated byte in the pre-defined/MPR value for all the bytes in the DDR burst (burst length 4 or 8). If the comparison fails then it stores the low read boundary of the associated byte for each byte at MPRDDLHWST0/1[HW_RD_DL_LOW#] . If the comparison passes then MMDC repeats steps 9-11. If all read data comparisons fail then the MMDC advances to the next step
13. The MMDC start seeking the upper boundary and sets the read delay line absolute offset of each byte to the initial value + 1 as determined at step 4 and issue measurement process of the read delay-line to update itself with the new value
14. MMDC resets the rd fifo (to the inverted pre-defined value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1
15. MMDC drives read command to the DDR external devices and waits 16 or 32 cycles (according to MPRDDLHWCTL[HW_RD_DL_CMP_CYC]) assuming that the data has arrived from the DDR device
16. MMDC compares the read data byte to the associated byte in the pre-defined/MPR value for all the bytes in the DDR burst (burst length 4 or 8). If the comparison fails then it stores the upper read boundary of the associated byte for each byte at MPRDDLHWST0/1[HW_RD_DL_UP#] . If the comparison passes then MMDC increments the read delay line absolute offset of each byte by 1 (i.e. MPRDDLCTL[RD_DL_ABS_OFFSET#]) and issue measurement process of the read delay-line to update itself with the new value.
17. If all read data comparisons fail then the MMDC advances to the next step. otherwise, MMDC repeats steps 14-16.
18. After the MMDC finds the window boundary (lower and upper) of each read data byte then it stores the average between lower and upper boundaries at the associated MPRDDLCTL[RD_DL_ABS_OFFSET#] and issue measurement process of the read delay-line to update itself with the new value.
19. MMDC indicates that the read data calibration had finished by setting MPRDDLHWCTL[HW_RD_DL_EN] = 0

20. Exit the DDR device from MPR/DQ calibration mode through MRS/MRW commands

44.11.4.2 SW Read Calibration

- There are two modes of operations:
- Calibration with the MPR (Multi Purpose Register)/DQ calibration(LPDDR2)
- Calibration with MMDC pre-defined values

44.11.4.2.1 Calibration with MPR(DDR3)/DQ calibration(LPDDR2)

The following steps should be executed:

1. Precharge all active banks (Can be done through MDSCR) as required by the standard.
2. Enter the DDR device into MPR/DQ calibration mode through MRS/MRW commands
3. Configure the MMDC to work with MPR/DQ calibration mode by asserting MPPDCMPR2[MPR_CMP]
4. Make sure that the initial value that is configured in the read delay line absolute offset of each byte (i.e. MPRDDLCTL[RD_DL_ABS_OFFSET#]) will place the read DQS somewhere inside the read DQ window

44.11.4.2.2 Calibration with pre-defined value

In case pre-defined mode is used, i.e. MPPDCMPR2[MPR_CMP] is cleared, then the following steps should be executed:

1. Precharge all active banks (Can be done through MDSCR) as requried by the standard.
2. Configure the pre-defined value, which reflects the value that will be written and compared through the read calibration, to MPPDCMPR1[PDV1, PDV2]
3. Issue write access (with any legal DDR address) to external DDR device.
4. Make sure that the initial value that is configured in the read delay line absolute offset of each byte (i.e. MPRDDLCTL[RD_DL_ABS_OFFSET#]) will place the read DQS somewhere inside the read DQ window

   The following steps will be executed manually by SW for both modes (MPR/DQ calibration and Pre-defined value):

5. Force the delay line to measure itself and to issue the requested read delay by configuring MPMUR[FRC_MSR] = 1

6. Wait 16 DDR cycles till the read delay-line is updated with the absolute delay value for all bytes
7. Issue read command (with any legal DDR address) from the external DDR device
8. Compare the read data byte to the associated byte in the pre-defined/MPR value for all the bytes in the DDR burst (burst length 4 or 8). If the comparison fails then it indicates that the initial read DQS isn't inside the read DQ window. If the comparison passes then advance to next step.
9. Reset the rd fifo (to the inverted pre-defined/MPR value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1
10. Decrement the read delay line absolute offset of each byte by 1 (i.e. MPRDDLCTL[RD_DL_ABS_OFFSET#])
11. Force the delay line to measure itself and to issue the requested read delay by configuring MPMUR[FRC_MSR] = 1
12. Issue read command (with the legal DDR address chosen in step 7) from the external DDR device and waits 16 or 32 cycles (according to MPRDDLHWCTL[HW_RD_DL_CMP_CYC]) assuming that the data has arrived from the DDR device
13. Compare the read data byte to the associated byte in the pre-defined/MPR value for all the bytes in the DDR burst (burst length 4 or 8). If the comparison fails then it is needed to store the low read boundary of the associated byte at of each byte . If the comparison passes then repeat steps 9-12. If all read data comparisons fail then advance to the next step.
14. Start seeking the upper boundary and set the read delay line absolute offset of each byte to the initial value + 1 as determined at step 4
15. Force the delay line to measure itself and to issue the requested read delay by configuring MPMUR[FRC_MSR] = 1
16. Resets the rd fifo (to the inverted pre-defined/MPR value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1
17. Issue read command (with the legal DDR address chosen in step 7) from the external DDR device
18. Compare the read data byte to the associated byte in the pre-defined/MPR value for all the bytes in the DDR burst (burst length 4 or 8). If the comparison fails then it is needed to store the upper read boundary of the associated byte at of each byte. If the comparison passes then increment the read delay line absolute offset of each byte by 1 (i.e. MPRDDLCTL[RD_DL_ABS_OFFSET#])
19. Force the delay line to measure itself and to issue the requested read delay by configuring MPMUR[FRC_MSR] = 1
20. If all read data comparisons fail then advance to the next step, else repeat steps 16-19
21. After finding the window boundary (lower and upper) of each read data byte then calculate the average between lower and upper boundaries and store the associated average at MPRDDLCTL[RD_DL_ABS_OFFSET#]

22. Force the delay line to measure itself and to issue the requested read delay by configuring MPMUR[FRC_MSR] = 1
23. Exit the DDR device from MPR/DQ calibration mode through MRS/MRW commands.

44.11.5 Write Calibration

The write calibration is used to adjust the write DQS with write data byte. It is assumed that the read calibration process is completed prior to the write calibration.

NOTE

In DDR3_x64 and LP2_1ch_x64 modes, the activation of the calibration is done by setting MPWRDLHWCTL0[HW_WR_DL_EN] at address 0xBASE0_0864

In LP2_2ch_x16, LP2_2ch_x32 the activation of the calibration of each channel is done by setting MPWRDLHWCTL0[HW_WR_DL_EN] at address 0xBASE0_0864 and MPWRDLHWCTL0[HW_WR_DL_EN] at address 0xBASE1_0864 respectively

44.11.5.1 HW (automatic) Write Calibration

The following steps should be executed:

1. Make sure that the initial value that is configured in the write delay line absolute offset of each byte (i.e. MPWRDLCTL[WR_DL_ABS_OFFSET#]) will place the write DQS somewhere inside the write DQ window
2. Configure the pre-defined value, which reflects the value that will be written and compared through the write calibration, to MPPDCMPR1[PDV1, PDV2]
3. Assert MPWRDLHWCTL0[HW_WR_DL_EN]

   The following steps will be executed automatically:

4. MMDC waits till the write delay-line is updated with the absolute delay value for all bytes at MPWRDCTL[WR_DL_ABS_OFFSET#]
5. MMDC drives write command to the external DDR devices (to bank 0 address 0) and waits 16 or 32 cycles (according to MPWRDLHWCTL[HW_WR_DL_CMP_CYC]) assuming that the data has arrived to the DDR device.
6. MMDC drives read command to the same address from the external DDR

7. MMDC compares the read data byte to the associated byte in the pre-defined value for all the bytes in the DDR burst (burst length 4 or 8). If the comparison fails then it indicates that the initial write DQS isn't inside the write DQ window and the MMDC generates an error for the associated byte at MPWRDLHWCTL[HW_WR_DL_ERR#] . If the comparison passes then MMDC advances to next step.
8. MMDC resets the rd fifo (to the inverted pre-defined value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1
9. MMDC decrements the write delay line absolute offset of each byte by 1 (i.e. MPWRDLCTL[WR_DL_ABS_OFFSET#]) and issue measurement process of the write delay-line to update itself with the new value.
10. MMDC drives write command to the external DDR devices (to bank 0 address 0) and waits 16 or 32 cycles (according to MPWRDLHWCTL[HW_WR_DL_CMP_CYC]) assuming that the data has arrived to the DDR device
11. MMDC drives read command to the same address from the external DDR
12. MMDC compares the read data byte to the associated byte in the pre-defined value for all the bytes in the DDR burst (burst length 4 or 8). If the comparison fails then it stores the low write boundary of the associated byte of each byte at MPWRDLHWST0/1[HW_WR_DL_LOW#] . If the comparison passes then MMDC repeats steps 8-11. If all data comparisons fail then the MMDC advances to the next step
13. The MMDC start seeking the upper boundary and sets the write delay line absolute offset of each byte to the initial value + 1 as determined at step 4 and issue measurement process of the write delay-line to update itself with the new value
14. MMDC resets the rd fifo (to the inverted pre-defined value) and its pointers by setting MPDGCTRL[RST_RD_FIFO] = 1
15. MMDC drives write command to the external DDR devices (to bank 0 address 0) and waits 16 or 32 cycles (according to MPWRDLHWCTL[HW_WR_DL_CMP_CYC]) assuming that the data has arrived to the DDR device.
16. MMDC drives read command to the same address from the external DDR
17. MMDC compares the read data byte to the associated byte in the pre-defined value for all the bytes in the DDR burst (burst length 4 or 8). If the comparison fails then it stores the upper write boundary of the associated byte of each byte at MPWRDLHWST0/1[HW_WR_DL_UP#] . If the comparison passes then MMDC increments the write delay line absolute offset of each byte by 1 (i.e. MPWRDLCTL[WR_DL_ABS_OFFSET#]) and issue measurement process of the write delay-line to update itself with the new value.
18. MMDC repeats steps 14-17. If all data comparisons fail then the MMDC advances to the next step
19. .After the MMDC finds the window boundary (lower and upper) of each write data byte then it stores the average between lower and upper boundaries at the associated

MPWRDLCTL[WR_DL_ABS_OFFSET#] and issue measurement process of the write delay-line to update itself with the new value.

20. MMDC indicates that the write data calibration had finished by setting MPWRDLHWCTL[HW_WR_DL_EN] = 0

44.11.5.2 SW Write Calibration

The following steps should be executed:

NOTE

It is recommended to perform the write calibration using the HW method. The SW method is provided for debug purposes only.

1. Make sure that the initial value that is configured in the write delay line absolute offset of each byte (i.e. MPWRDLCTL[WR_DL_ABS_OFFSET#]) will place the write DQS somewhere inside the write DQ window
2. Configure the pre-defined value, which reflects the value that will be written and compared through the write calibration, to MPPDCMPR1[PDV1, PDV2]
3. Force the delay line to measure itself and to issue the requested write delay by configuring MPMUR[FRC_MSR] = 1
4. Wait 16 DDR cycles till the write delay-line is updated with the absolute delay value for all bytes
5. Issue write command to any legal DDR address of the external DDR device
6. Issue read command, to the address written previously, from the external DDR device
7. Compare the read data byte to the associated byte in the pre-defined value for all bytes in the DDR burst (burst length 4 or 8). If the comparison fails then it indicates that the initial write DQS isn't inside the write DQ window. If the comparison passes then advance to next step.
8. MMDC resets the rd fifo (to the inverted pre-defined value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1
9. Decrement the write delay line absolute offset of each byte by 1 (i.e. MPWRDLCTL[WR_DL_ABS_OFFSET#])
10. Force the delay line to measure itself and to issue the requested write delay by configuring MPMUR[FRC_MSR] = 1
11. Issue write command to any legal DDR address of the external DDR device
12. Issue read command, to the address written previously, from the external DDR device
13. Compare the read data byte to the associated byte in the pre-defined value for all the bytes in the DDR burst (burst length 4 or 8). If the comparison fails then it is needed

to store the low write boundary of the associated byte of each byte at MPWRDLHWST0/1[HW_WR_DL_LOW#] . If the comparison passes then repeat steps 8-12. If all data comparisons fail then advance to the next step.
14. Start seeking the upper boundary and set the write delay line absolute offset of each byte to the initial value + 1
15. Force the delay line to measure itself and to issue the requested write delay by configuring MPMUR[FRC_MSR] = 1
16. Reset the rd fifo (to the inverted pre-defined value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1
17. Issue write command to any legal DDR address of the external DDR device
18. Issue read command, to the address written previously, from the external DDR device
19. Compare the read data byte to the associated byte in the pre-defined value for all the bytes in the DDR burst (burst length 4 or 8). If the comparison fails then it is needed to store the upper write boundary of the associated byte of each byte at MPWRDLHWST0/1[HW_WR_DL_UP#]. If the comparison passes then increment the write delay line absolute offset of each byte by 1.
20. Force the delay line to measure itself and to issue the requested write delay by configuring MPMUR[FRC_MSR] = 1
21. If all read data comparisons fail then advance to the next step else repeat steps 16-20.
22. .After finding the window boundary (lower and upper) of each write data byte then calculate the average between lower and upper boundaries and store the associated average at MPWRDLCTL[WR_DL_ABS_OFFSET#]
23. Force the delay line to measure itself and to issue the requested write delay by configuring MPMUR[FRC_MSR] = 1

44.11.6 Write leveling Calibration

The write leveling calibration can generate a delay between the clock and the associate DQS of up to 3 cycles as following: (WL_DL_ABS_OFFSET/256*cycle) + (WL_HC_DEL*half cycle) + (WL_CYC_DEL*cycle).

Write leveling calibration can be executed automatically(HW) or manually (SW).

The automatic calibration process can only detect the optimal DQS to clock delay to within 1 cycle. In extreme cases in which the DDR3 memory is placed far from the microcontroller (long address/command/clock trace lengths), the skew between the DQS and clock may exceed 1 cycle. If this is the case, it is the user's responsibility to both understand that their design causes the DQS to clock skew to exceed 1 cycle and to indicate this manually in the MPWLDECTRL0/1[WL_CYC_DEL#]. It is highly recommended to keep the DDR3 memory as close to the microcontroller as possible,

especially in embedded system designs. When using fly-by topology, the user should calculate the PCB flight time of the clock signal to the furthest placed DDR3 memory to ensure less than 1 cycle skew between DQS and clock.

NOTE

In LPDDR2 mode Write-leveling calibration should be disabled.

In DDR3_x64 mode activation of the calibration is done by setting MPWLGCR[HW_WL_EN] at address 0xBASE1_0808

NOTE

It is essential to route the first bit in each data byte group (D0, D8, D16, D24, D32, D40, D48, D56) from the DDR3 memory to the same data bus bits on the controller. The DDR3 memory outputs the state of the DRAM clock during the write leveling calibration. If any of these are not routed properly, the controller will have no information regarding the state of the DRAM clock during calibration. In previous designs in which write leveling calibration was not performed, the board designer would often swap data bits within each byte group to make the data bus routing cleaner and less susceptible to noise and impedance mismatch. This can still be done with DDR3 so long as the requirement of properly routing D0, D8, D16, D24, D32, D40, D48, D56 is maintained.

44.11.6.1 Hardware Write Leveling Calibration

The following steps should be executed:

1. Configure the external DDR device to enter write leveling mode through MRS command
2. Activate the DQS output enable by setting MDSCR[WL_EN]
3. Active automatic calibration by setting MPWLGCR[HW_WL_EN]

   The following steps will be executed automatically by the MMDC:

4. MMDC enters write leveling mode, counts 25 + 15 cycles and drives the DQS pads as output while the DQ pads will remain inputs. In parallel the MMDC configures the write leveling delay line to "0" (i.e. MPWLDECTRL0[WL_DL_ABS_OFFSET#] = 0) and issue measurement process of the writ-leveling delay-line to update itself with the new value
5. MMDC drives one DQS pulse to the DDR external device

6. MMDC waits 16 cycles (to guarantee that the DQ prime data is stable) and samples the associated prime DQ bit (for example for DQS1 the MMDC samples DQ[8])
7. MMDC increments the write leveling delay line by 1/8 cycle and perform measurement process in order to load the updated value to the associated delay-line
8. MMDC repeates steps 5-7 till the write leveling delay is 1 cycle
9. MMDC checks the 8 bit prime DQ results for each DQS and finds the first transition from 0 to 1. If no transition is found then the MMDC indicates an error at MPWLGCR[HW_WL_ERR#]
10. MMDC stores the value that issues the last "0" on the prime DQ before the transition and loads it to the write leveling delay-line. The MMDC initiates a fine-tune process by incrementing the delay-line values by 1 step (which is 1/256 part of a cycle) till detecting the most accurate transition from 0 to 1
11. Upon completion of this process the MMDC de-asserts the MPWLGCR[HW_WL_EN] and update the most accurate value of the delay-line at the associated MPWLDECTRL#[WL_DL_ABS_OFFSET#]
12. MMDC perform measurement process in order to load the most accurate value to the associated delay-line
13. User should issue MRS command to exit write leveling mode
14. The user should read the results of the associated delay-line at MPWLDECTRL#[WL_DL_ABS_OFFSET#] and in case the user estimates that the reasonable delay may be above 1 cycle then the user should indicate it at MPWLDECTRL#[WL_CYC_DEL#]. Moreover the user should indicate it in MDMISC[WALAT] field. For example, if the result of the write leveling calibration is 100/256 parts of a cycle, but the user estimates that the delay is above 2 cycles then MPWLDECTRL#[WL_CYC_DEL#] should be configured to 2, so the total delay will be 2 and 100/256 parts of a cycle
15. Return the DQS output enable to functional mode by deasserting MDSCR[WL_EN]

44.11.6.2 SW Write Leveling Calibration

The following steps should be executed:

NOTE

It is recommended to perform the write calibration using the HW method. The SW method is provided for debug purposes only.

1. Configure the external DDR device to enter write leveling mode through MRS command
2. Activate the DQS output enable by setting MDSCR[WL_EN]

3. Set the write-leveling delay-line offset to "0" by configuring MPWLDECTRL0[WL_DL_ABS_OFFSET#] = 0
4. Force the delay line to measure itself and to issue the requested write-leveling delay by configuring MPMUR[FRC_MSR] = 1
5. Activate SW write-leveling calibration and issue one DQS pulse by setting MPWLGCR[SW_WL_EN] = 1 together with MPWLGCR[SW_WL_CNT_EN] = 1
6. Issue an IP read command from MPWLGCR. If MPWLGCR[SW_WL_EN] = 0 then the SW write-leveling result is valid at MPWLGCR[WL_SW_RES#].
7. Increment the write leveling delay line by 1/8 cycle (i.e add 0x20 to {MPWLDECTRL0[WL_HC_DEL#],MPWLDECTRL0[WL_DL_ABS_OFFSET#]})
8. Force the delay line to measure itself and to issue the requested write-leveling delay by configuring MPMUR[FRC_MSR] = 1
9. Activate SW write-leveling calibration and issue one DQS pulse by setting MPWLGCR[SW_WL_EN] = 1
10. Repeate steps 6-9 till the edge of CK was detected (i.e the write-leveling result switched from "0" to "1")
11. Store the value that issues the last "0" on the prime DQ before the transition and load it to the write leveling delay-line and start fine tuning process to detect the exact switch from "0" to "1"
12. Force the delay line to measure itself and to issue the requested write-leveling delay by configuring MPMUR[FRC_MSR] = 1
13. Activate SW write-leveling calibration and issue one DQS pulse by setting MPWLGCR[SW_WL_EN] = 1
14. Issue a IP read command from MPWLGCR. If MPWLGCR[SW_WL_EN] = 0 then the SW write-leveling result is valid at MPWLGCR[WL_SW_RES#].
15. Increment the write leveling delay line by 1 step (i.e add 0x01 to MPWLDECTRL0[WL_DL_ABS_OFFSET#])
16. Force the delay line to measure itself and to issue the requested write-leveling delay by configuring MPMUR[FRC_MSR] = 1
17. Issue an IP read command from MPWLGCR. If MPWLGCR[SW_WL_EN] = 0 then the SW write-leveling result is valid at MPWLGCR[WL_SW_RES#].
18. Activate SW write-leveling calibration and issue one DQS pulse by setting MPWLGCR[SW_WL_EN] = 1
19. Repeates step 15-18 till the exact edge of CK was detected (i.e the write-leveling result switched from "0" to "1")
20. Issue MRS command to exit write leveling mode
21. Return the DQS output enable to functional mode by deasserting MDSCR[WL_EN]

44.11.7 Write fine tuning

Write fine tuning is an additional circuit that provides the ability to fine tune the timing of each dq/dm bits (relative to dqs) by up to +/-100 ps.

This is done by reducing the delay between the wl_dqs by 100 ps and adding a configurable delay of up to 200 ps (6 delay units of around 30-35 ps each) for each DQ/DM output. The delay can be configured independently for each DQ/DM. The calibration of this mechanism can be done only by writing & reading data from the memory. Controlled by register MPWRDQBY#DL.

44.11.8 Read fine tuning

Read fine tuning is an additional circuit that provides the ability to fine tune the timing of each coming dq bits (relative to coming dqs) by up to +/-100 ps.

This is done by reducing the delay between the incoming rd_dqs by 100 ps and adding a configurable delay of up to 200 ps (6 delay units of around 30-35 ps each) for each DQ input. The delay can be configured independently for each DQ. The calibration of this mechanism can be done only by writing & reading data from the memory. Controlled by register MPRDDQBY#DL.

44.12 MMDC Memory Map/Register Definition

MMDC may be configured to several modes of operation. See Table 44-12 .

Table 44-12. MMDC - Modes of Operation

Mode of Operation	Abbreviation
DDR3 x16	DDR3_x16
DDR3 x32	DDR3_x32
DDR3 x64	DDR3_x64
LPDDR2 2-channels x16	LP2_2ch_x16
LPDDR2 2-channels x32	LP2_2ch_x32

NOTE

In case of LPDDR2 2-channels mode while the external memory devices are exactly the same for both channels then it is recommended to configure the registers located at

0x021B_0000 - 0x021B_0440 and 0x021B_4000 - 0x021B_4440 to the same values

In case of DDR3_x64 then IP port1 is used only to configure the parameters related to the calibration process of Byte4 - Byte7 at addresses 0x021B_4808 - 0x021B_48C0 as shown in Table 44-13 .

Table 44-13. MMDC - Modes of Operation

Mode of Operation	Associated Memory Map
DDR3 x16, x32	0x021B_0000 - 0x021B_08C4
DDR3 x64	0x021B_0000 - 0x021B_08C4 0x021B_4808 - 0x021B_48C0
LPDDR2 2-channels x16, x32	0x021B_0000 - 0x021B_08C4 0x021B_4808 - 0x021B_48C0

Table 44-14 shows the register mode of operations.

Table 44-14. Register Mode of Operations

Registers	1-Channel Mode of Operations	2-Channel Mode of Operations
MMDC Core Control Register	all	LP2_2ch_x16, LP2_2ch_x32
MMDC Core Power Down Control Register		all
MMDC Core ODT Timing Control Register		LP2_2ch_x16, LP2_2ch_x32
MMDC Core Timing Configuration Register 0		
MMDC Core Timing Configuration Register 1		
MMDC Core Timing Configuration Register 2		
MMDC Core Miscellaneous Register		
MMDC Core Special Command Register		
MMDC Core Refresh Control Register		
Reserved		all
Reserved		
MMDC Core Read/Write Command Delay Register		LP2_2ch_x16, LP2_2ch_x32
MMDC Core Out of Reset Delays Register		
MMDC Core MRR Data Register	LP2_2ch_x16, LP2_2ch_x32	LP2_2ch_x16, LP2_2ch_x32
MMDC Core Timing Configuration Register 3		LP2_2ch_x16, LP2_2ch_x32
MMDC Core MR4 Derating Register		
MMDC Core Address Space Partition Register	all	
MMDC Core AXI Reordering Control Regsiter		
MMDC Core Power Saving Control and Status Register		
MMDC Core Exclusive ID Monitor Register0		
MMDC Core Exclusive ID Monitor Register1		

Table continues on the next page...

Table 44-14. Register Mode of Operations (continued)

Registers	1-Channel Mode of Operations	2-Channel Mode of Operations
MMDC Core Debug and Profiling Control Register 0		
MMDC Core Debug and Profiling Control Register 1		
MMDC Core Debug and Profiling Status Register 0		
MMDC Core Debug and Profiling Status Register 1		
MMDC Core Debug and Profiling Status Register 2		
MMDC Core Debug and Profiling Status Register 3		
MMDC Core Debug and Profiling Status Register 4		
MMDC Core Debug and Profiling Status Register 5		
MMDC Core Step By Step Address Register		
MMDC Core Step By Step Address Attributes Register		
Reserved		all
MMDC Core General Purpose Register		LP2_2ch_x16, LP2_2ch_x32
MMDC PHY ZQ HW control register		all
MMDC PHY ZQ SW control register		
MMDC PHY Write Leveling Configuration and Error Status Register		DDR3_x64, LP2_2ch_x16, LP2_2ch_x32
MMDC PHY Write Leveling Delay Control Register 0		
MMDC PHY Write Leveling Delay Control Register 1		DDR3_x64, LP2_2ch_x32
MMDC PHY Write Leveling delay-line Status Register		DDR3_x64, LP2_2ch_x16, LP2_2ch_x32
MMDC PHY ODT control register	DDR3_x16, DDR3_x32, DDR3_x64	DDR3_x64
MMDC PHY Read DQ Byte0 Delay Register	all	DDR3_x64, LP2_2ch_x16, LP2_2ch_x32
MMDC PHY Read DQ Byte1 Delay Register		
MMDC PHY Read DQ Byte2 Delay Register		DDR3_x64, LP2_2ch_x32
MMDC PHY Read DQ Byte3 Delay Register		
MMDC PHY Write DQ Byte0 Delay Register		DDR3_x64, LP2_2ch_x16, LP2_2ch_x32
MMDC PHY Write DQ Byte1 Delay Register		
MMDC PHY Write DQ Byte2 Delay Register		DDR3_x64, LP2_2ch_x32
MMDC PHY Write DQ Byte3 Delay Register		
MMDC PHY Read DQS Gating Control Register 0	DDR3_x16, DDR3_x32, DDR3_x64	DDR3_x64
MMDC PHY Read DQS Gating Control Register 1		
MMDC PHY Read DQS Gating delay-line Status Register		
MMDC PHY Read delay-lines Configuration Register	all	DDR3_x64, LP2_2ch_x16, LP2_2ch_x32
MMDC PHY Read delay-lines Status Register		
MMDC PHY Write delay-lines Configuration Register		
MMDC PHY Write delay-lines Status Register		
MMDC PHY CK Control Register		LP2_2ch_x16, LP2_2ch_x32
MMDC ZQ LPDDR2 HW Control Register	LP2_2ch_x16, LP2_2ch_x32	

Table continues on the next page...

Table 44-14. Register Mode of Operations (continued)

Registers	1-Channel Mode of Operations	2-Channel Mode of Operations
MMDC PHY Read Delay HW Calibration Control Register	all	DDR3_x64, LP2_2ch_x16, LP2_2ch_x32
MMDC PHY Write Delay HW Calibration Control Register		
MMDC PHY Read Delay HW Calibration Status Register 0		
MMDC PHY Read Delay HW Calibration Status Register 1		DDR3_x64, LP2_2ch_x32
MMDC PHY Write Delay HW Calibration Status Register 0		DDR3_x64, LP2_2ch_x16, LP2_2ch_x32
MMDC PHY Write Delay HW Calibration Status Register 1		DDR3_x64, LP2_2ch_x32
MMDC PHY Write Leveling HW Error Register		DDR3_x64, LP2_2ch_x16, LP2_2ch_x32
MMDC PHY Read DQS Gating HW Status Register 0	DDR3_x16, DDR3_x32, DDR3_x64	DDR3_x64
MMDC PHY Read DQS Gating HW Status Register 1		
MMDC PHY Read DQS Gating HW Status Register 2		
MMDC PHY Read DQS Gating HW Status Register 3		
MMDC PHY Pre-defined Compare Register 1	all	LP2_2ch_x16, LP2_2ch_x32
MMDC PHY Pre-defined Compare and CA delay-line Configuration Register		
MMDC PHY SW Dummy Access Register		DDR3_x64, LP2_2ch_x16, LP2_2ch_x32
MMDC PHY SW Dummy Read Data Register 0		
MMDC PHY SW Dummy Read Data Register 1		
MMDC PHY SW Dummy Read Data Register 2		
MMDC PHY SW Dummy Read Data Register 3		
MMDC PHY SW Dummy Read Data Register 4		
MMDC PHY SW Dummy Read Data Register 5		
MMDC PHY SW Dummy Read Data Register 6		
MMDC PHY SW Dummy Read Data Register 7		
MMDC PHY Measure Unit Register		LP2_2ch_x16, LP2_2ch_x32
MMDC Write CA delay-line controller	LP2_2ch_x16, LP2_2ch_x32	
MMDC Duty Cycle Control Register	all	DDR3_x64, LP2_2ch_x16, LP2_2ch_x32

Table 44-15 shows the maximum AXI address space for the main modes of operation.

Table 44-15. MMDC - Maximum AXI Address Space Per Operation Mode

Mode of Operation	Maximum AXI Address Space	Comment
LP2_2ch_x32	AXI port0: 0x8000_0000 - 0xFFFF_FFFFF	Up to 2GB for LPDDR2 Channel 0
	AXI port1: 0x1000_0000 - 0x7FFF_FFFFF	Up to 1.75GB for LPDDR2 Channel 1
DDR3_x64	AXI port0: 0x1000_0000 - 0xFFFF_FFFFF	All address space is associated with AXI port0

The Memory Map is shown below.

MMDC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21B_0000	MMDC Core Control Register (MMDC1_MDCTL)	32	R/W	0311_0000h	44.12.1/ 3897
21B_0004	MMDC Core Power Down Control Register (MMDC1_MDPDC)	32	R/W	0003_0012h	44.12.2/ 3899
21B_0008	MMDC Core ODT Timing Control Register (MMDC1_MDOTC)	32	R/W	1227_2000h	44.12.3/ 3901
21B_000C	MMDC Core Timing Configuration Register 0 (MMDC1_MDCFG0)	32	R/W	3236_22D3h	44.12.4/ 3903
21B_0010	MMDC Core Timing Configuration Register 1 (MMDC1_MDCFG1)	32	R/W	B6B1_8A23h	44.12.5/ 3905
21B_0014	MMDC Core Timing Configuration Register 2 (MMDC1_MDCFG2)	32	R/W	00C7_0092h	44.12.6/ 3907
21B_0018	MMDC Core Miscellaneous Register (MMDC1_MDMISC)	32	R/W	0000_1600h	44.12.7/ 3909
21B_001C	MMDC Core Special Command Register (MMDC1_MDSCR)	32	R/W	0000_0000h	44.12.8/ 3912
21B_0020	MMDC Core Refresh Control Register (MMDC1_MDREF)	32	R/W	0000_C000h	44.12.9/ 3915
21B_002C	MMDC Core Read/Write Command Delay Register (MMDC1_MDRWD)	32	R/W	0F9F_26D2h	44.12.10/ 3918
21B_0030	MMDC Core Out of Reset Delays Register (MMDC1_MDOR)	32	R/W	009F_0E0Eh	44.12.11/ 3920
21B_0034	MMDC Core MRR Data Register (MMDC1_MDMRR)	32	R	0000_0000h	44.12.12/ 3921
21B_0038	MMDC Core Timing Configuration Register 3 (MMDC1_MDCFG3LP)	32	R/W	0000_0000h	44.12.13/ 3922
21B_003C	MMDC Core MR4 Derating Register (MMDC1_MDMR4)	32	R/W	0000_0000h	44.12.14/ 3923
21B_0040	MMDC Core Address Space Partition Register (MMDC1_MDASP)	32	R/W	0000_003Fh	44.12.15/ 3925
21B_0400	MMDC Core AXI Reordering Control Regsiter (MMDC1_MAARCR)	32	R/W	5142_01F0h	44.12.16/ 3926
21B_0404	MMDC Core Power Saving Control and Status Register (MMDC1_MAPSR)	32	R/W	0000_1007h	44.12.17/ 3928
21B_0408	MMDC Core Exclusive ID Monitor Register0 (MMDC1_MAEXIDR0)	32	R/W	0020_0000h	44.12.18/ 3931
21B_040C	MMDC Core Exclusive ID Monitor Register1 (MMDC1_MAEXIDR1)	32	R/W	0060_0040h	44.12.19/ 3931
21B_0410	MMDC Core Debug and Profiling Control Register 0 (MMDC1_MADPCR0)	32	R/W	0000_0000h	44.12.20/ 3932

Table continues on the next page...

MMDC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21B_0414	MMDC Core Debug and Profiling Control Register 1 (MMDC1_MADPCR1)	32	R/W	0000_0000h	44.12.21/ 3933
21B_0418	MMDC Core Debug and Profiling Status Register 0 (MMDC1_MADPSR0)	32	R	0000_0000h	44.12.22/ 3934
21B_041C	MMDC Core Debug and Profiling Status Register 1 (MMDC1_MADPSR1)	32	R	0000_0000h	44.12.23/ 3934
21B_0420	MMDC Core Debug and Profiling Status Register 2 (MMDC1_MADPSR2)	32	R	0000_0000h	44.12.24/ 3935
21B_0424	MMDC Core Debug and Profiling Status Register 3 (MMDC1_MADPSR3)	32	R	0000_0000h	44.12.25/ 3935
21B_0428	MMDC Core Debug and Profiling Status Register 4 (MMDC1_MADPSR4)	32	R	0000_0000h	44.12.26/ 3936
21B_042C	MMDC Core Debug and Profiling Status Register 5 (MMDC1_MADPSR5)	32	R	0000_0000h	44.12.27/ 3937
21B_0430	MMDC Core Step By Step Address Register (MMDC1_MASBS0)	32	R	0000_0000h	44.12.28/ 3937
21B_0434	MMDC Core Step By Step Address Attributes Register (MMDC1_MASBS1)	32	R	0000_0000h	44.12.29/ 3938
21B_0440	MMDC Core General Purpose Register (MMDC1_MAGENP)	32	R/W	0000_0000h	44.12.30/ 3939
21B_0800	MMDC PHY ZQ HW control register (MMDC1_MPZQHWCTRL)	32	R/W	A138_0000h	44.12.31/ 3939
21B_0804	MMDC PHY ZQ SW control register (MMDC1_MPZQSWCTRL)	32	R/W	0000_0000h	44.12.32/ 3942
21B_0808	MMDC PHY Write Leveling Configuration and Error Status Register (MMDC1_MPWLGCR)	32	R/W	0000_0000h	44.12.33/ 3944
21B_080C	MMDC PHY Write Leveling Delay Control Register 0 (MMDC1_MPWLDECTRL0)	32	R/W	0000_0000h	44.12.34/ 3947
21B_0810	MMDC PHY Write Leveling Delay Control Register 1 (MMDC1_MPWLDECTRL1)	32	R/W	0000_0000h	44.12.35/ 3949
21B_0814	MMDC PHY Write Leveling delay-line Status Register (MMDC1_MPWLDLST)	32	R	0000_0000h	44.12.36/ 3951
21B_0818	MMDC PHY ODT control register (MMDC1_MPODTCTRL)	32	R/W	0000_0000h	44.12.37/ 3953
21B_081C	MMDC PHY Read DQ Byte0 Delay Register (MMDC1_MPRDDQBY0DL)	32	R/W	0000_0000h	44.12.38/ 3955
21B_0820	MMDC PHY Read DQ Byte1 Delay Register (MMDC1_MPRDDQBY1DL)	32	R/W	0000_0000h	44.12.39/ 3958
21B_0824	MMDC PHY Read DQ Byte2 Delay Register (MMDC1_MPRDDQBY2DL)	32	R/W	0000_0000h	44.12.40/ 3961
21B_0828	MMDC PHY Read DQ Byte3 Delay Register (MMDC1_MPRDDQBY3DL)	32	R/W	0000_0000h	44.12.41/ 3964
21B_082C	MMDC PHY Write DQ Byte0 Delay Register (MMDC1_MPWRDQBY0DL)	32	R/W	0000_0000h	44.12.42/ 3966

Table continues on the next page...

MMDC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21B_0830	MMDC PHY Write DQ Byte1 Delay Register (MMDC1_MPWRDQBY1DL)	32	R/W	0000_0000h	44.12.43/ 3968
21B_0834	MMDC PHY Write DQ Byte2 Delay Register (MMDC1_MPWRDQBY2DL)	32	R/W	0000_0000h	44.12.44/ 3971
21B_0838	MMDC PHY Write DQ Byte3 Delay Register (MMDC1_MPWRDQBY3DL)	32	R/W	0000_0000h	44.12.45/ 3973
21B_083C	MMDC PHY Read DQS Gating Control Register 0 (MMDC1_MPDGCTRL0)	32	R/W	0000_0000h	44.12.46/ 3975
21B_0840	MMDC PHY Read DQS Gating Control Register 1 (MMDC1_MPDGCTRL1)	32	R/W	0000_0000h	44.12.47/ 3978
21B_0844	MMDC PHY Read DQS Gating delay-line Status Register (MMDC1_MPDGDLST0)	32	R	0000_0000h	44.12.48/ 3979
21B_0848	MMDC PHY Read delay-lines Configuration Register (MMDC1_MPRDDLCTL)	32	R/W	4040_4040h	44.12.49/ 3981
21B_084C	MMDC PHY Read delay-lines Status Register (MMDC1_MPRDDLST)	32	R	0000_0000h	44.12.50/ 3982
21B_0850	MMDC PHY Write delay-lines Configuration Register (MMDC1_MPWRDLCTL)	32	R/W	4040_4040h	44.12.51/ 3983
21B_0854	MMDC PHY Write delay-lines Status Register (MMDC1_MPWRDLST)	32	R	0000_0000h	44.12.52/ 3985
21B_0858	MMDC PHY CK Control Register (MMDC1_MPSDCTRL)	32	R/W	0000_0000h	44.12.53/ 3986
21B_085C	MMDC ZQ LPDDR2 HW Control Register (MMDC1_MPZQLP2CTL)	32	R/W	1B5F_0109h	44.12.54/ 3987
21B_0860	MMDC PHY Read Delay HW Calibration Control Register (MMDC1_MPRDDLHWCTL)	32	R/W	0000_0000h	44.12.55/ 3988
21B_0864	MMDC PHY Write Delay HW Calibration Control Register (MMDC1_MPWRDLHWCTL)	32	R/W	0000_0000h	44.12.56/ 3990
21B_0868	MMDC PHY Read Delay HW Calibration Status Register 0 (MMDC1_MPRDDLHWST0)	32	R	0000_0000h	44.12.57/ 3992
21B_086C	MMDC PHY Read Delay HW Calibration Status Register 1 (MMDC1_MPRDDLHWST1)	32	R	0000_0000h	44.12.58/ 3993
21B_0870	MMDC PHY Write Delay HW Calibration Status Register 0 (MMDC1_MPWRDLHWST0)	32	R	0000_0000h	44.12.59/ 3994
21B_0874	MMDC PHY Write Delay HW Calibration Status Register 1 (MMDC1_MPWRDLHWST1)	32	R	0000_0000h	44.12.60/ 3995
21B_0878	MMDC PHY Write Leveling HW Error Register (MMDC1_MPWLHWERR)	32	R/W	0000_0000h	44.12.61/ 3996
21B_087C	MMDC PHY Read DQS Gating HW Status Register 0 (MMDC1_MPDGHWST0)	32	R	0000_0000h	44.12.62/ 3996
21B_0880	MMDC PHY Read DQS Gating HW Status Register 1 (MMDC1_MPDGHWST1)	32	R	0000_0000h	44.12.63/ 3997
21B_0884	MMDC PHY Read DQS Gating HW Status Register 2 (MMDC1_MPDGHWST2)	32	R	0000_0000h	44.12.64/ 3998

Table continues on the next page...

MMDC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21B_0888	MMDC PHY Read DQS Gating HW Status Register 3 (MMDC1_MPDGHWST3)	32	R	0000_0000h	44.12.65/ 3998
21B_088C	MMDC PHY Pre-defined Compare Register 1 (MMDC1_MPPDCMPR1)	32	R/W	0000_0000h	44.12.66/ 3999
21B_0890	MMDC PHY Pre-defined Compare and CA delay-line Configuration Register (MMDC1_MPPDCMPR2)	32	R/W	0040_0000h	44.12.67/ 4000
21B_0894	MMDC PHY SW Dummy Access Register (MMDC1_MPSWDAR0)	32	R/W	0000_0000h	44.12.68/ 4001
21B_0898	MMDC PHY SW Dummy Read Data Register 0 (MMDC1_MPSWDRDR0)	32	R	FFFF_FFFFh	44.12.69/ 4003
21B_089C	MMDC PHY SW Dummy Read Data Register 1 (MMDC1_MPSWDRDR1)	32	R	FFFF_FFFFh	44.12.70/ 4004
21B_08A0	MMDC PHY SW Dummy Read Data Register 2 (MMDC1_MPSWDRDR2)	32	R	FFFF_FFFFh	44.12.71/ 4004
21B_08A4	MMDC PHY SW Dummy Read Data Register 3 (MMDC1_MPSWDRDR3)	32	R	FFFF_FFFFh	44.12.72/ 4005
21B_08A8	MMDC PHY SW Dummy Read Data Register 4 (MMDC1_MPSWDRDR4)	32	R	FFFF_FFFFh	44.12.73/ 4005
21B_08AC	MMDC PHY SW Dummy Read Data Register 5 (MMDC1_MPSWDRDR5)	32	R	FFFF_FFFFh	44.12.74/ 4006
21B_08B0	MMDC PHY SW Dummy Read Data Register 6 (MMDC1_MPSWDRDR6)	32	R	FFFF_FFFFh	44.12.75/ 4006
21B_08B4	MMDC PHY SW Dummy Read Data Register 7 (MMDC1_MPSWDRDR7)	32	R	FFFF_FFFFh	44.12.76/ 4007
21B_08B8	MMDC PHY Measure Unit Register (MMDC1_MPMUR0)	32	R/W	0000_0000h	44.12.77/ 4007
21B_08BC	MMDC Write CA delay-line controller (MMDC1_MPWRCADL)	32	R/W	0000_0000h	44.12.78/ 4008
21B_08C0	MMDC Duty Cycle Control Register (MMDC1_MPDCCR)	32	R	2492_2492h	44.12.79/ 4010
21B_4000	MMDC Core Control Register (MMDC2_MDCTL)	32	R/W	0311_0000h	44.12.1/ 3897
21B_4004	MMDC Core Power Down Control Register (MMDC2_MDPDC)	32	R/W	0003_0012h	44.12.2/ 3899
21B_4008	MMDC Core ODT Timing Control Register (MMDC2_MDOTC)	32	R/W	1227_2000h	44.12.3/ 3901
21B_400C	MMDC Core Timing Configuration Register 0 (MMDC2_MDCFG0)	32	R/W	3236_22D3h	44.12.4/ 3903
21B_4010	MMDC Core Timing Configuration Register 1 (MMDC2_MDCFG1)	32	R/W	B6B1_8A23h	44.12.5/ 3905
21B_4014	MMDC Core Timing Configuration Register 2 (MMDC2_MDCFG2)	32	R/W	00C7_0092h	44.12.6/ 3907
21B_4018	MMDC Core Miscellaneous Register (MMDC2_MDMISC)	32	R/W	0000_1600h	44.12.7/ 3909

Table continues on the next page...

MMDC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21B_401C	MMDC Core Special Command Register (MMDC2_MDSCR)	32	R/W	0000_0000h	44.12.8/ 3912
21B_4020	MMDC Core Refresh Control Register (MMDC2_MDREF)	32	R/W	0000_C000h	44.12.9/ 3915
21B_402C	MMDC Core Read/Write Command Delay Register (MMDC2_MDRWD)	32	R/W	0F9F_26D2h	44.12.10/ 3918
21B_4030	MMDC Core Out of Reset Delays Register (MMDC2_MDOR)	32	R/W	009F_0E0Eh	44.12.11/ 3920
21B_4034	MMDC Core MRR Data Register (MMDC2_MDMRR)	32	R	0000_0000h	44.12.12/ 3921
21B_4038	MMDC Core Timing Configuration Register 3 (MMDC2_MDCFG3LP)	32	R/W	0000_0000h	44.12.13/ 3922
21B_403C	MMDC Core MR4 Derating Register (MMDC2_MDMR4)	32	R/W	0000_0000h	44.12.14/ 3923
21B_4040	MMDC Core Address Space Partition Register (MMDC2_MDASP)	32	R/W	0000_003Fh	44.12.15/ 3925
21B_4400	MMDC Core AXI Reordering Control Regsiter (MMDC2_MAARCR)	32	R/W	5142_01F0h	44.12.16/ 3926
21B_4404	MMDC Core Power Saving Control and Status Register (MMDC2_MAPSR)	32	R/W	0000_1007h	44.12.17/ 3928
21B_4408	MMDC Core Exclusive ID Monitor Register0 (MMDC2_MAEXIDR0)	32	R/W	0020_0000h	44.12.18/ 3931
21B_440C	MMDC Core Exclusive ID Monitor Register1 (MMDC2_MAEXIDR1)	32	R/W	0060_0040h	44.12.19/ 3931
21B_4410	MMDC Core Debug and Profiling Control Register 0 (MMDC2_MADPCR0)	32	R/W	0000_0000h	44.12.20/ 3932
21B_4414	MMDC Core Debug and Profiling Control Register 1 (MMDC2_MADPCR1)	32	R/W	0000_0000h	44.12.21/ 3933
21B_4418	MMDC Core Debug and Profiling Status Register 0 (MMDC2_MADPSR0)	32	R	0000_0000h	44.12.22/ 3934
21B_441C	MMDC Core Debug and Profiling Status Register 1 (MMDC2_MADPSR1)	32	R	0000_0000h	44.12.23/ 3934
21B_4420	MMDC Core Debug and Profiling Status Register 2 (MMDC2_MADPSR2)	32	R	0000_0000h	44.12.24/ 3935
21B_4424	MMDC Core Debug and Profiling Status Register 3 (MMDC2_MADPSR3)	32	R	0000_0000h	44.12.25/ 3935
21B_4428	MMDC Core Debug and Profiling Status Register 4 (MMDC2_MADPSR4)	32	R	0000_0000h	44.12.26/ 3936
21B_442C	MMDC Core Debug and Profiling Status Register 5 (MMDC2_MADPSR5)	32	R	0000_0000h	44.12.27/ 3937
21B_4430	MMDC Core Step By Step Address Register (MMDC2_MASBS0)	32	R	0000_0000h	44.12.28/ 3937
21B_4434	MMDC Core Step By Step Address Attributes Register (MMDC2_MASBS1)	32	R	0000_0000h	44.12.29/ 3938

Table continues on the next page...

MMDC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21B_4440	MMDC Core General Purpose Register (MMDC2_MAGENP)	32	R/W	0000_0000h	44.12.30/ 3939
21B_4800	MMDC PHY ZQ HW control register (MMDC2_MPZQHWCTRL)	32	R/W	A138_0000h	44.12.31/ 3939
21B_4804	MMDC PHY ZQ SW control register (MMDC2_MPZQSWCTRL)	32	R/W	0000_0000h	44.12.32/ 3942
21B_4808	MMDC PHY Write Leveling Configuration and Error Status Register (MMDC2_MPWLGCR)	32	R/W	0000_0000h	44.12.33/ 3944
21B_480C	MMDC PHY Write Leveling Delay Control Register 0 (MMDC2_MPWLDECTRL0)	32	R/W	0000_0000h	44.12.34/ 3947
21B_4810	MMDC PHY Write Leveling Delay Control Register 1 (MMDC2_MPWLDECTRL1)	32	R/W	0000_0000h	44.12.35/ 3949
21B_4814	MMDC PHY Write Leveling delay-line Status Register (MMDC2_MPWLDLST)	32	R	0000_0000h	44.12.36/ 3951
21B_4818	MMDC PHY ODT control register (MMDC2_MPODTCTRL)	32	R/W	0000_0000h	44.12.37/ 3953
21B_481C	MMDC PHY Read DQ Byte0 Delay Register (MMDC2_MPRDDQBY0DL)	32	R/W	0000_0000h	44.12.38/ 3955
21B_4820	MMDC PHY Read DQ Byte1 Delay Register (MMDC2_MPRDDQBY1DL)	32	R/W	0000_0000h	44.12.39/ 3958
21B_4824	MMDC PHY Read DQ Byte2 Delay Register (MMDC2_MPRDDQBY2DL)	32	R/W	0000_0000h	44.12.40/ 3961
21B_4828	MMDC PHY Read DQ Byte3 Delay Register (MMDC2_MPRDDQBY3DL)	32	R/W	0000_0000h	44.12.41/ 3964
21B_482C	MMDC PHY Write DQ Byte0 Delay Register (MMDC2_MPWRDQBY0DL)	32	R/W	0000_0000h	44.12.42/ 3966
21B_4830	MMDC PHY Write DQ Byte1 Delay Register (MMDC2_MPWRDQBY1DL)	32	R/W	0000_0000h	44.12.43/ 3968
21B_4834	MMDC PHY Write DQ Byte2 Delay Register (MMDC2_MPWRDQBY2DL)	32	R/W	0000_0000h	44.12.44/ 3971
21B_4838	MMDC PHY Write DQ Byte3 Delay Register (MMDC2_MPWRDQBY3DL)	32	R/W	0000_0000h	44.12.45/ 3973
21B_483C	MMDC PHY Read DQS Gating Control Register 0 (MMDC2_MPDGCTRL0)	32	R/W	0000_0000h	44.12.46/ 3975
21B_4840	MMDC PHY Read DQS Gating Control Register 1 (MMDC2_MPDGCTRL1)	32	R/W	0000_0000h	44.12.47/ 3978
21B_4844	MMDC PHY Read DQS Gating delay-line Status Register (MMDC2_MPDGDLST0)	32	R	0000_0000h	44.12.48/ 3979
21B_4848	MMDC PHY Read delay-lines Configuration Register (MMDC2_MPRDDLCTL)	32	R/W	4040_4040h	44.12.49/ 3981
21B_484C	MMDC PHY Read delay-lines Status Register (MMDC2_MPRDDLST)	32	R	0000_0000h	44.12.50/ 3982
21B_4850	MMDC PHY Write delay-lines Configuration Register (MMDC2_MPWRDLCTL)	32	R/W	4040_4040h	44.12.51/ 3983

Table continues on the next page...

MMDC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21B_4854	MMDC PHY Write delay-lines Status Register (MMDC2_MPWRDLST)	32	R	0000_0000h	44.12.52/ 3985
21B_4858	MMDC PHY CK Control Register (MMDC2_MPSDCTRL)	32	R/W	0000_0000h	44.12.53/ 3986
21B_485C	MMDC ZQ LPDDR2 HW Control Register (MMDC2_MPZQLP2CTL)	32	R/W	1B5F_0109h	44.12.54/ 3987
21B_4860	MMDC PHY Read Delay HW Calibration Control Register (MMDC2_MPRDDLHWCTL)	32	R/W	0000_0000h	44.12.55/ 3988
21B_4864	MMDC PHY Write Delay HW Calibration Control Register (MMDC2_MPWRDLHWCTL)	32	R/W	0000_0000h	44.12.56/ 3990
21B_4868	MMDC PHY Read Delay HW Calibration Status Register 0 (MMDC2_MPRDDLHWST0)	32	R	0000_0000h	44.12.57/ 3992
21B_486C	MMDC PHY Read Delay HW Calibration Status Register 1 (MMDC2_MPRDDLHWST1)	32	R	0000_0000h	44.12.58/ 3993
21B_4870	MMDC PHY Write Delay HW Calibration Status Register 0 (MMDC2_MPWRDLHWST0)	32	R	0000_0000h	44.12.59/ 3994
21B_4874	MMDC PHY Write Delay HW Calibration Status Register 1 (MMDC2_MPWRDLHWST1)	32	R	0000_0000h	44.12.60/ 3995
21B_4878	MMDC PHY Write Leveling HW Error Register (MMDC2_MPWLHWERR)	32	R/W	0000_0000h	44.12.61/ 3996
21B_487C	MMDC PHY Read DQS Gating HW Status Register 0 (MMDC2_MPDGHWST0)	32	R	0000_0000h	44.12.62/ 3996
21B_4880	MMDC PHY Read DQS Gating HW Status Register 1 (MMDC2_MPDGHWST1)	32	R	0000_0000h	44.12.63/ 3997
21B_4884	MMDC PHY Read DQS Gating HW Status Register 2 (MMDC2_MPDGHWST2)	32	R	0000_0000h	44.12.64/ 3998
21B_4888	MMDC PHY Read DQS Gating HW Status Register 3 (MMDC2_MPDGHWST3)	32	R	0000_0000h	44.12.65/ 3998
21B_488C	MMDC PHY Pre-defined Compare Register 1 (MMDC2_MPPDCMPR1)	32	R/W	0000_0000h	44.12.66/ 3999
21B_4890	MMDC PHY Pre-defined Compare and CA delay-line Configuration Register (MMDC2_MPPDCMPR2)	32	R/W	0040_0000h	44.12.67/ 4000
21B_4894	MMDC PHY SW Dummy Access Register (MMDC2_MPSWDAR0)	32	R/W	0000_0000h	44.12.68/ 4001
21B_4898	MMDC PHY SW Dummy Read Data Register 0 (MMDC2_MPSWDRDR0)	32	R	FFFF_FFFFh	44.12.69/ 4003
21B_489C	MMDC PHY SW Dummy Read Data Register 1 (MMDC2_MPSWDRDR1)	32	R	FFFF_FFFFh	44.12.70/ 4004
21B_48A0	MMDC PHY SW Dummy Read Data Register 2 (MMDC2_MPSWDRDR2)	32	R	FFFF_FFFFh	44.12.71/ 4004
21B_48A4	MMDC PHY SW Dummy Read Data Register 3 (MMDC2_MPSWDRDR3)	32	R	FFFF_FFFFh	44.12.72/ 4005
21B_48A8	MMDC PHY SW Dummy Read Data Register 4 (MMDC2_MPSWDRDR4)	32	R	FFFF_FFFFh	44.12.73/ 4005

Table continues on the next page...

MMDC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21B_48AC	MMDC PHY SW Dummy Read Data Register 5 (MMDC2_MPSWDRDR5)	32	R	FFFF_FFFFh	44.12.74/ 4006
21B_48B0	MMDC PHY SW Dummy Read Data Register 6 (MMDC2_MPSWDRDR6)	32	R	FFFF_FFFFh	44.12.75/ 4006
21B_48B4	MMDC PHY SW Dummy Read Data Register 7 (MMDC2_MPSWDRDR7)	32	R	FFFF_FFFFh	44.12.76/ 4007
21B_48B8	MMDC PHY Measure Unit Register (MMDC2_MPMUR0)	32	R/W	0000_0000h	44.12.77/ 4007
21B_48BC	MMDC Write CA delay-line controller (MMDC2_MPWRCADL)	32	R/W	0000_0000h	44.12.78/ 4008
21B_48C0	MMDC Duty Cycle Control Register (MMDC2_MPDCCR)	32	R	2492_2492h	44.12.79/ 4010

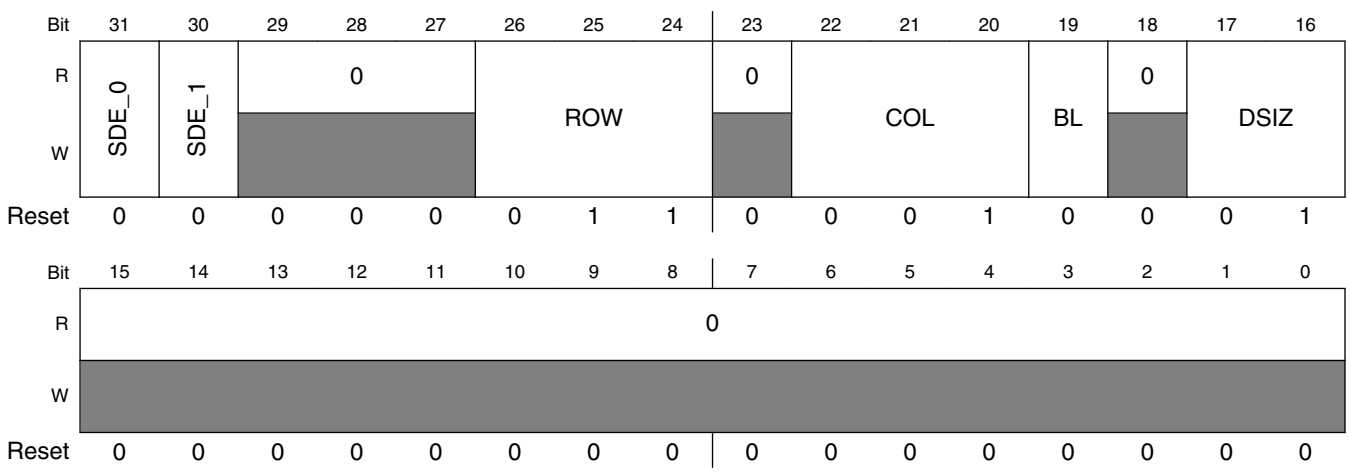
44.12.1 MMDC Core Control Register (MMDCx_MDCTL)

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 0h offset



MMDCx_MDCTL field descriptions

Field	Description
31 SDE_0	MMDC Enable CS0. This bit enables/disables accesses from the MMDC toward Chip Select 0. The reset value of this bit is "0" (i.e No clocks and clock enable will be driven to the memory). At the enabling point the MMDC will perform an initialization process (including a delay on RESET and/or CKE) for both chip selects. The initialization length depends on the configured memory type.

Table continues on the next page...

MMDCx_MDCTL field descriptions (continued)

Field	Description
	0 Disabled 1 Enabled
30 SDE_1	MMDC Enable CS1. This bit enables/disables accesses from the MMDC toward Chip Select 1. The reset value of this bit is "0" (i.e No clocks and clock enable will be driven to the memory). At the enabling point the MMDC will perform an initialization process (including a delay on RESET and/or CKE) for both chip selects. The initialization length depends on the configured memory type. 0 Disabled 1 Enabled
29–27 Reserved	This read-only field is reserved and always has the value 0.
26–24 ROW	Row Address Width. This field specifies the number of row addresses used by the memory array. It will affect the way an incoming address will be decoded. Settings 110-111 are reserved 000 11 bits Row 001 12 bits Row 010 13 bits Row 011 14 bits Row 100 15 bits Row 101 16 bits Row
23 Reserved	This read-only field is reserved and always has the value 0.
22–20 COL	Column Address Width. This field specifies the number of column addresses used by the memory array. It will determine how an incoming address will be decoded. 0x0 9 bits column 0x1 10 bits column 0x2 11 bits column 0x3 8 bits column 0x4 12 bits column 0x5-0xF Reserved
19 BL	Burst Length. This field determines the burst length of the DDR device. In LPDDR2 mode the MMDC supports burst length 4. In DDR3 mode the MMDC supports burst length 8. 0 Burst Length 4 is used 1 Burst Length 8 is used
18 Reserved	This read-only field is reserved and always has the value 0.
17–16 DSIZ	DDR data bus size. This field determines the size of the data bus of the DDR memory 0 16-bit data bus 1 32-bit data bus 2 64-bit data bus 3 Reserved —

Table continues on the next page...

MMDCx_MDCTL field descriptions (continued)

Field	Description
Reserved	This read-only field is reserved and always has the value 0.

44.12.2 MMDC Core Power Down Control Register (MMDCx_MDPDC)

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: LP2_2ch_x16, LP2_2ch_x32

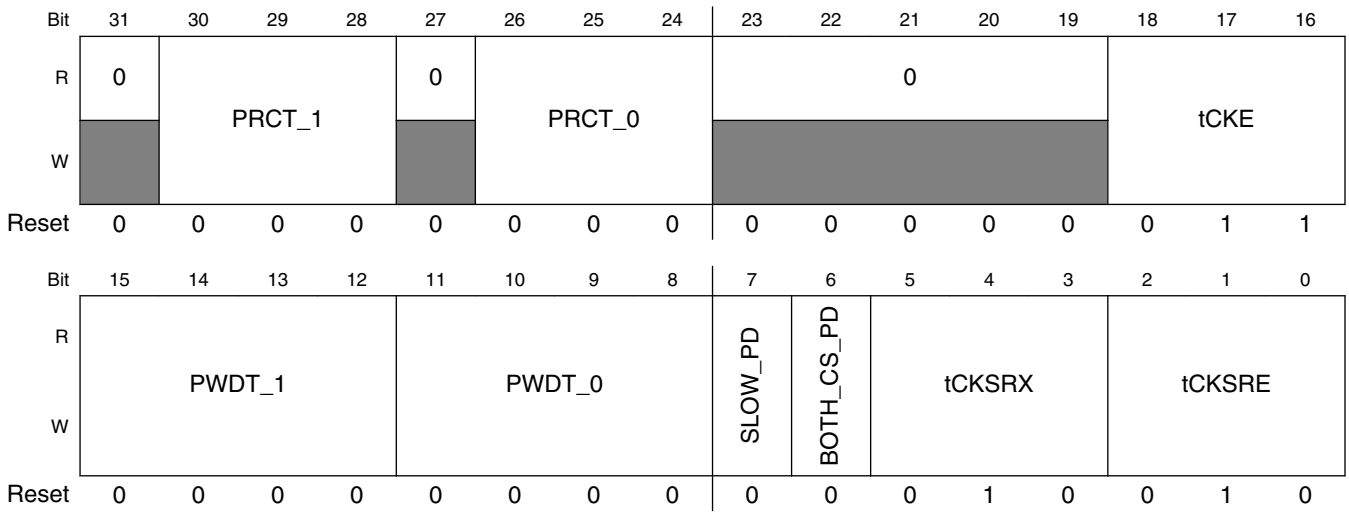
Table 44-107. PRCT field encoding

PRCT[2:0]	Precharge Timer
000	Disabled (Bit field reset value)
001	2 clocks
010	4 clocks
011	8 clocks
100	16 clocks
101	32 clocks
110	64 clocks
111	128 clocks

Table 44-108. PWDT field encoding

PWDT[3:0]	Power Down Time-out
0000	Disabled (bit field reset value)
0001	16 cycles
0010	32 cycles
0011	64 cycles
0100	128 cycles
0101	256 cycles
0110	512 cycles
0111	1024 cycles
1000	2048 cycles
1001	4096 cycles
1010	8196 cycles
1011	16384 cycles
1100	32768 cycles
1101-1111	Reserved

Address: Base address + 4h offset



MMDCx_MDPDC field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30–28 PRCT_1	Precharge Timer - Chip Select 1. This field determines the amount of idle cycle for which chip select 1 will be automatically precharged. The amount of cycles are determined according to the PRCT Field Encoding table above.
27 Reserved	This read-only field is reserved and always has the value 0.
26–24 PRCT_0	Precharge Timer - Chip Select 0. This field determines the amount of idle cycle for which chip select 0 will be automatically precharged. The amount of cycles are determined according to the table below.
23–19 Reserved	This read-only field is reserved and always has the value 0.
18–16 tCKE	CKE minimum pulse width. This field determines the minimum pulse width of CKE. 0x0 1 cycle 0x1 2 cycles 0x6 7 cycles 0x7 8 cycles
15–12 PWDT_1	Power Down Timer - Chip Select 1. This field determines the amount of idle cycle for which chip select 1 will be automatically get into precharge/active power down. The amount of cycles are determined according to the PWDT Field Encoding table above.
11–8 PWDT_0	Power Down Timer - Chip Select 0. This field determines the amount of idle cycle for which chip select 0 will be automatically get into precharge/active power down. The amount of cycles are determined according to the PWDT Field Encoding table above.
7 SLOW_PD	Slow/fast power down. In DDR3 mode this field is referred to slow precharge power-down.

Table continues on the next page...

MMDCx_MDPDC field descriptions (continued)

Field	Description
	In LPDDR2 mode this field is not relevant. **NOTE:** Memory should be configured the same. 0 Fast mode. 1 Slow mode.
6 BOTH_CS_PD	Parallel power down entry to both chip selects. When power down timer is used for both chip-selects (i.e PWDT_0 and PWDT1 don't equal "0") , then if this bit is enabled, the MMDC will enter power down only if the amount of idle cycles of both chip selects was obtained. 0 Each chip select can enter power down independently according to its configuration. 1 Chip selects can enter power down only if the amount of idle cycles of both chip selects was obtained.
5–3 tCKSRX	Valid clock cycles before self-refresh exit. This field determines the amount of clock cycles before self-refresh exit 0x0 0 cycle 0x1 1 cycles 0x6 6 cycles 0x7 7 cycles
tCKSRE	Valid clock cycles after self-refresh entry. This field determines the amount of clock cycles after self-refresh entry 0x0 0 cycle 0x1 1 cycles 0x6 6cycles 0x7 7cycles

44.12.3 MMDC Core ODT Timing Control Register (MMDCx_MDOTC)

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: LP2_2ch_x16, LP2_2ch_x32

For further information see ODT Configuration .

Address: Base address + 8h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0		tAOFPD			tAONPD			tANPD				tAXPD			
W																
Reset	0	0	0	1	0	0	1	0	0	0	1	0	0	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	tODTLon			0				tODT_idle_off				0			
W																
Reset	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDCx_MDOTC field descriptions

Field	Description
31–30 Reserved	This read-only field is reserved and always has the value 0.
29–27 tAOFPD	Asynchronous RTT turn-off delay (power down with DLL frozen). This field determines the time between termination cuircuit starts to turn off the ODT resistance till termination has reached high impedance. This field is not relevant in LPDDR2 mode. 0x0 1 cycle 0x1 2 cycles 0x6 7 cycles 0x7 8 cycles
26–24 tAONPD	Asynchronous RTT turn-on delay (power down with DLL frozen). This field determines the time between termination cuircuit gets out of high impedance and begins to turn on till ODT resistance are fully on. This field is not relevant in LPDDR2 mode. 0x0 1 cycle 0x1 2 cycles 0x6 7 cycles 0x7 8 cycles
23–20 tANPD	Asynchronous ODT to power down entry delay. In DDR3 should be set to tCWL-1 This field is not relevant in LPDDR2 mode. 0x0 1 clock 0x1 2 clocks 0x2 3 clocks 0xE 15 clocks 0xF 16 clocks
19–16 tAXPD	Asynchronous ODT to power down exit delay. In DDR3 should be set to tCWL-1 This field is not relevant in LPDDR2 mode. 0x0 1 clock 0x1 2 clocks 0x2 3 clocks 0xE 15 clocks 0xF 16 clocks
15 Reserved	This read-only field is reserved and always has the value 0.
14–12 tODTLon	ODT turn on latency. This field determines the delay between ODT signal and the associated RTT, where according to JEDEC standard it equals WL(write latency) - 2. Therefore, the value that is configured to tODTLon field should correspond the value that is configured to MDCGFG1[tCWL] In LPDDR2 this field is not relevant. 0x0 - 0x1 Reserved

Table continues on the next page...

MMDCx_MDOTC field descriptions (continued)

Field	Description
	0x2 2 cycles 0x3 3 cycles 0x4 4 cycles 0x5 5 cycles 0x6 6 cycles 0x7 Reserved
11–9 Reserved	This read-only field is reserved and always has the value 0.
8–4 tODT_idle_off	ODT turn off latency. This field determines the Idle period before turning memory ODT off. This field is not relevant in LPDDR2 mode. 0x0 0 cycle (turned off at the earliest possible time) 0x1 1 cycle 0x2 2 cycles 0x1E 30 cycles 0x1F 31 cycles
Reserved	This read-only field is reserved and always has the value 0.

44.12.4 MMDC Core Timing Configuration Register 0 (MMDCx_MDCFG0)

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: LP2_2ch_x16, LP2_2ch_x32

Address: Base address + Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	tRFC								tXS								tXP			tXPDLL				tFAW							tCL			
Reset	0	0	1	1	0	0	1	0	0	0	1	1	0	1	1	0	0	0	1	0	0	0	1	0	1	1	0	1	0	0	1	1

MMDCx_MDCFG0 field descriptions

Field	Description
31–24 tRFC	Refresh command to Active or Refresh command time. See DDR3 SDRAM Specification JESD79-3E (July 2010) and LPDDR2 SDRAM Specification JESD209-2B (February 2010) for a detailed description of this parameter. 0x0 1 clock 0x1 2 clocks 0x2 3 clocks 0xFE 255 clocks 0xFF 256 clocks

Table continues on the next page...

MMDCx_MDCFG0 field descriptions (continued)

Field	Description
23–16 tXS	Exit self refresh to non READ command. In LPDDR2 it is called tXSR, self-refresh exit to next valid command delay. See DDR3 SDRAM Specification JESD79-3E (July 2010) and LPDDR2 SDRAM Specification JESD209-2B (February 2010) for a detailed description of this parameter. 0x0 - 0x15 reserved 0x16 23 clocks 0x17 24 clocks 0xFE 255 clocks 0xFF 256 clocks
15–13 tXP	Exit power down with DLL-on to any valid command. Exit power down with DLL-frozen to commands not requiring a locked DLL In LPDDR2 mode this field is referred to Exit power-down to next valid command delay. See DDR3 SDRAM Specification JESD79-3E (July 2010) and LPDDR2 SDRAM Specification JESD209-2B (February 2010) for a detailed description of this parameter. 0x0 1 cycle 0x1 2 cycles 0x6 7 cycles 0x7 8 cycles
12–9 tXPDLL	Exit precharge power down with DLL frozen to commands requiring DLL. This field is not relevant in LPDDR2 mode. See DDR3 SDRAM Specification JESD79-3E (July 2010) and LPDDR2 SDRAM Specification JESD209-2B (February 2010) for a detailed description of this parameter. 0x0 1 clock 0x1 2 clocks 0x2 3 clocks 0xE 15 clocks 0xF 16 clocks
8–4 tFAW	Four Active Window (all banks). See DDR3 SDRAM Specification JESD79-3E (July 2010) and LPDDR2 SDRAM Specification JESD209-2B (February 2010) for a detailed description of this parameter. 0x0 1 clock 0x1 2 clocks 0x2 3 clocks 0x1E 31 clocks 0x1F 32 clocks
tCL	CAS Read Latency. In DDR3 mode this field is referred to CL. In LPDDR2 mode this field is referred to RL. **NOTE:** In LPDDR2 mode only the RL/WL pairs are allowed as specified in MR2 register See DDR3 SDRAM Specification JESD79-3E (July 2010) and LPDDR2 SDRAM Specification JESD209-2B (February 2010) for a detailed description of this parameter.

Table continues on the next page...

MMDCx_MDCFG0 field descriptions (continued)

Field	Description
	0x0 3 cycles 0x1 4 cycles 0x2 5 cycles 0x3 6 cycles 0x4 7 cycles 0x5 8 cycles 0x6 9 cycles 0x7 10 cycles 0x8 11 cycles 0x9 - 0xF Reserved

44.12.5 MMDC Core Timing Configuration Register 1 (MMDCx_MDCFG1)

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 10h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	tRCD			tRP			tRC					tRAS				
Reset	1	0	1	1	0	1	1	0	1	0	1	1	0	0	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	tRPA	0			tWR				tMRD			0		tCWL		
Reset	1	0	0	0	1	0	1	0	0	0	1	0	0	0	1	1

MMDCx_MDCFG1 field descriptions

Field	Description
31–29 tRCD	Active command to internal read or write delay time (same bank). (This field is valid only for DDR3 memories) In LPDDR2 mode this parameter should be configured at tRCD_LP. See DDR3 SDRAM Specification JESD79-3E (July 2010) and LPDDR2 SDRAM Specification JESD209-2B (February 2010) for a detailed description of this parameter. 0x0 1 clock 0x1 2 clocks 0x2 3 clocks 0x3 4 clocks 0x4 5 clocks 0x5 6 clocks

Table continues on the next page...

MMDCx_MDCFG1 field descriptions (continued)

Field	Description
	0x6 7 clocks 0x7 8 clocks
28–26 tRP	Precharge command period (same bank). (This field is valid only for DDR3 memories) IIn LPDDR2 mode this parameter should be configured at tRPpb_LP. See DDR3 SDRAM Specification JESD79-3E (July 2010) and LPDDR2 SDRAM Specification JESD209-2B (February 2010) for a detailed description of this parameter. 0x0 1 clock 0x1 2 clocks 0x2 3 clocks 0x3 4 clocks 0x4 5 clocks 0x5 6 clocks 0x6 7 clocks 0x7 8 clocks
25–21 tRC	Active to Active or Refresh command period (same bank). (This field is valid only for DDR3 memories) In LPDDR2 mode this parameter should be configured at tRC_LP. See DDR3 SDRAM Specification JESD79-3E (July 2010) and LPDDR2 SDRAM Specification JESD209-2B (February 2010) for a detailed description of this parameter. 0x0 1 clock 0x1 2 clocks 0x2 3 clocks 0x1E 31 clocks 0x1F 32 clocks
20–16 tRAS	Active to Precharge command period (same bank). See DDR3 SDRAM Specification JESD79-3E (July 2010) and LPDDR2 SDRAM Specification JESD209-2B (February 2010) for a detailed description of this parameter. 0x0 1 clock 0x1 2 clocks 0x2 3 clocks 0x1E 31 clocks 0x1F Reserved
15 tRPA	Precharge-all command period. (This field is valid only for DDR3 memories) In LPDDR2 mode this parameter should be configured at tRPab_LP. See DDR3 SDRAM Specification JESD79-3E (July 2010) and LPDDR2 SDRAM Specification JESD209-2B (February 2010) for a detailed description of this parameter. 0 Will be equal to: tRP. 1 Will be equal to: tRP+1.

Table continues on the next page...

MMDCx_MDCFG1 field descriptions (continued)

Field	Description
14–12 Reserved	This read-only field is reserved and always has the value 0.
11–9 tWR	WRITE recovery time (same bank). See DDR3 SDRAM Specification JESD79-3E (July 2010) and LPDDR2 SDRAM Specification JESD209-2B (February 2010) for a detailed description of this parameter. 0x0 1cycle 0x1 2cycles 0x2 3cycles 0x3 4cycles 0x4 5cycles 0x5 6cycles 0x6 7cycles 0x7 8 cycles
8–5 tMRD	Mode Register Set command cycle (all banks). In DDR3 mode this field shoud be set to max (tMRD,tMOD). In LPDDR2 mode this field should be set to max(tMRR,tMRW) See DDR3 SDRAM Specification JESD79-3E (July 2010) and LPDDR2 SDRAM Specification JESD209-2B (February 2010) for a detailed description of this parameter. 0x0 1 clock 0x1 2 clocks 0x2 3 clocks 0xE 15 clocks 0xF 16 clocks
4–3 Reserved	This read-only field is reserved and always has the value 0.
tCWL	CAS Write Latency. In DDR3 mode this field is referred to CWL. In LPDDR2 mode this field is referred to WL. 0x0 2cycles (DDR3) , 1 cycle (LPDDR2) 0x1 3cycles (DDR3) , 2 cycles (LPDDR2) 0x2 4cycles (DDR3) , 3 cycles (LPDDR2) 0x3 5cycles (DDR3) , 4 cycles (LPDDR2) 0x4 6cycles (DDR3) , 5 cycles (LPDDR2) 0x5 7cycles (DDR3) , 6 cycles (LPDDR2) 0x6 8cycles (DDR3) , 7 cycles (LPDDR2) 0x7 Reserved

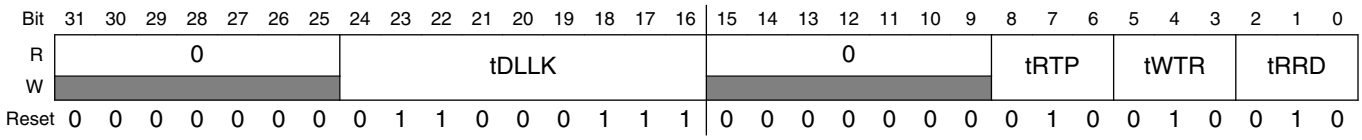
44.12.6 MMDC Core Timing Configuration Register 2 (MMDCx_MDCFG2)

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 14h offset



MMDCx_MDCFG2 field descriptions

Field	Description
31–25 Reserved	This read-only field is reserved and always has the value 0.
24–16 tDLLK	DLL locking time. This field is not relevant in LPDDR2 mode. See DDR3 SDRAM Specification JESD79-3E (July 2010) and LPDDR2 SDRAM Specification JESD209-2B (February 2010) for a detailed description of this parameter. 0x0 1 cycle. 0x1 2 cycles. 0x2 3 cycles. 0xC7 200 cycles 0x1FE 511 cycles. 0x1FF 512 cycles (JEDEC value for DDR3).
15–9 Reserved	This read-only field is reserved and always has the value 0.
8–6 tRTP	Internal READ command to Precharge command delay (same bank). See DDR3 SDRAM Specification JESD79-3E (July 2010) and LPDDR2 SDRAM Specification JESD209-2B (February 2010) for a detailed description of this parameter. 0x0 1cycle 0x1 2cycles 0x2 3cycles 0x3 4cycles 0x4 5cycles 0x5 6cycles 0x6 7cycles 0x7 8 cycles
5–3 tWTR	Internal WRITE to READ command delay (same bank). See DDR3 SDRAM Specification JESD79-3E (July 2010) and LPDDR2 SDRAM Specification JESD209-2B (February 2010) for a detailed description of this parameter. 0x0 1cycle 0x1 2cycles 0x2 3cycles 0x3 4cycles 0x4 5cycles 0x5 6cycles

Table continues on the next page...

MMDCx_MDCFG2 field descriptions (continued)

Field	Description
	0x6 7cycles 0x7 8 cycles
tRRD	Active to Active command period (all banks). See DDR3 SDRAM Specification JESD79-3E (July 2010) and LPDDR2 SDRAM Specification JESD209-2B (February 2010) for a detailed description of this parameter. 0x0 1cycle 0x1 2cycles 0x2 3cycles 0x3 4cycles 0x4 5cycles 0x5 6cycles 0x6 7cycles 0x7 Reserved

44.12.7 MMDC Core Miscellaneous Register (MMDCx_MDMISC)

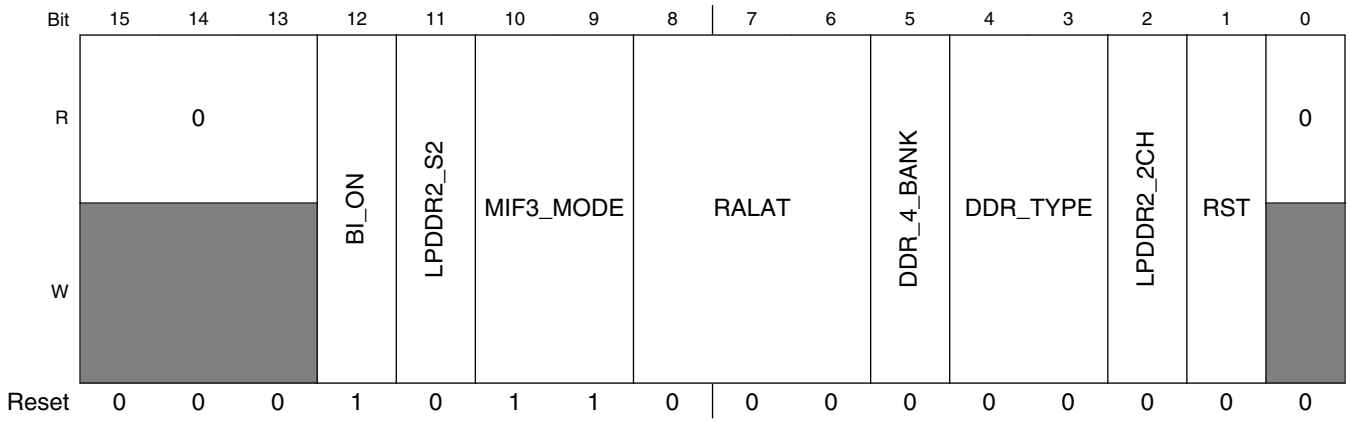
Supported Mode Of Operations:

For Channel 0: All

For Channel 1: LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 18h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CS0_RDY	CS1_RDY	0									CALIB_PER_CS	ADDR_MIRROR	LHD	WALAT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



MMDCx_MDMISC field descriptions

Field	Description
31 CS0_RDY	External status device on CS0. This is a read-only status bit, that indicates whether the external memory is in wake-up period. 0 Device in wake-up period. 1 Device is ready for initialization.
30 CS1_RDY	External status device on CS1. This is a read-only status bit, that indicates whether the external memory is in wake-up period. 0 Device in wake-up period. 1 Device is ready for initialization.
29–21 Reserved	This read-only field is reserved and always has the value 0.
20 CALIB_PER_CS	Number of chip-select for calibration process. This bit determines the chip-select index that the associated calibration is targetted to. Relevant for read, write, write leveling and read DQS gating calibrations 0 Calibration is targetted to CS0 1 Calibration is targetted to CS1
19 ADDR_MIRROR	Address mirroring. **NOTE:** This feature is not supported for LPDDR2 memories. But only for DDR3 memories. For further information see Address mirroring . 0 Address mirroring disabled. 1 Address mirroring enabled.
18 LHD	Latency hiding disable. This is a debug feature. When set to "1" the MMDC will handle one read/write access at a time. Meaning that the MMDC pipe-line will be limitted to 1 open access (next AXI address phase will be acknowledged if the current AXI data phase had finished) 0 Latency hiding on. 1 Latency hiding disable.
17–16 WALAT	Write Additional latency. In case the write-leveling calibration process indicates a delay of greater than one-eighth a clock cycle (between CK and any of the DQS strobe lines), then this field must be configured accordingly.

Table continues on the next page...

MMDCx_MDMISC field descriptions (continued)

Field	Description
	This field will add delay on the obe I/O control, which will compensate on the additional write leveling delay on DQS and prevent the DQS from being cropped. **NOTE:** The purpose of WALAT is to add time delay at the end of a burst write operation to ensure that the JEDEC time specification for Write Post Amble Delay (tWPST) is met (DQS strobe is held low at the end of a write burst for > 30% a clock cycle before it is released). If the value of any of the WL_DL_ABS_OFFSETn register fields are greater than '1F', WALAT should be set to '1' (cycle additional delay). WALAT should be further increased for any full cycle delays added by the WL_CYC_DELn register fields. 0x0 No additional latency required. 0x1 1 cycle additional delay 0x2 2 cycles additional delay 0x3 3 cycles additional delay
15–13 Reserved	This read-only field is reserved and always has the value 0.
12 BI_ON	Bank Interleaving On. This bit controls the organization of the bank, row and column address bits. For further information see Address decoding . 0 Banks are not interleaved, and address will be decoded as bank-row-column 1 Banks are interleaved, and address will be decoded as row-bank-column
11 LPDDR2_S2	LPDDR2 S2 device type indication. In case LPDDR2 device is used (DDR_TYPE = 0x1), this bit will indicate whether S2 or S4 device is used. This bit should be cleared in DDR3 mode 0x0 LPDDR2-S4 device is used. 0x1 LPDDR2-S2 device is used.
10–9 MIF3_MODE	Command prediction working mode. This field determines the level of command prediction that will be used by the MMDC 00 Disable prediction. 01 Enable prediction based on : Valid access on first pipe line stage. 10 Enable prediction based on: Valid access on first pipe line stage, Valid access on axi bus. 11 Enable prediction based on: Valid access on first pipe line stage, Valid access on axi bus, Next miss access from access queue.
8–6 RALAT	Read Additional Latency. This field determines the additional read latency which is added to CAS latency and internal delays for which the MMDC will retrieve the read data from the internal FIFO. This field is used to compensate on board/chip delays. **NOTE:** In LPDDR2 mode 2 extra cycles will be added internally in order to compensate tDQSCK delay. 0x0 no additional latency. 0x1 1 cycle additional latency. 0x2 2 cycles additional latency. 0x3 3 cycles additional latency. 0x4 4 cycles additional latency. 0x5 5 cycles additional latency. 0x6 6 cycles additional latency. 0x7 7 cycles additional latency.

Table continues on the next page...

MMDCx_MDMISC field descriptions (continued)

Field	Description
5 DDR_4_BANK	Number of banks per DDR device. When this bit is set to "1" then the MMDC will work with DDR device of 4 banks. 0 8 banks device is being used. (Default) 1 4 banks device is being used
4–3 DDR_TYPE	DDR TYPE. This field determines the type of the external DDR device. 0x0 DDR3 device is used. (Default) 0x1 LPDDR2 device is used. — — 0x2 Reserved. 0x3 Reserved.
2 LPDDR2_2CH	LPDDR2 2-channels mode. When this bit is set to "1" then dual channel mode is activated. This field should be cleared for DDR3 mode. 0 1-channel mode (DDR3) 1 2-channels mode (LPDDR2)
1 RST	Software Reset. When this bit is asserted then the internal FSMs and registers of the MMDC will be initialized. **NOTE:** This bit once asserted gets deasserted automatically. 0 Do nothing. 1 Assert reset to the MMDC.
0 Reserved	This read-only field is reserved and always has the value 0.

44.12.8 MMDC Core Special Command Register (MMDCx_MDSCR)

This register is used to issue special commands manually toward the external DDR device (such as load mode register, manual self refresh, manual precharge and so on). Every write to this register will be interpreted as a command, and a read from this register will show the last command that was executed.

Every write to this register will result in one special command, and the IP bus will assert ips_xfr_wait as long as the special command is being carried out.

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 1Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CMD_ADDR_MSB_MR_OP								CMD_ADDR_LSB_MR_ADDR							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CON_REQ	CON_ACK	0			MRR_READ_DATA_VALID	WL_EN	0		CMD			CMD_CS	CMD_BA		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDCx_MDSCR field descriptions

Field	Description
31–24 CMD_ADDR_ MSB_MR_OP	Command/Address MSB. This field indicates the MSB of the command/Address. In LPDDR2 this field indicates the MRW operand
23–16 CMD_ADDR_ LSB_MR_ADDR	Command/Address LSB. This field indicates the LSB of the command/Address In LPDDR2 this field indicates the MRR/MRW address
15 CON_REQ	Configuration request. When this bit is set then the MMDC will clean the pending AXI accesses and will prevent from further AXI accesses to be acknowledged. This field guarantee safe configuration (or change configuration) of the MMDC while no access is in process and prevents an unexpected behaviour. After setting this bit, it is needed to poll on CON_ACK until it is set to "1". When CON_ACK is asserted then configuration is permitted. After configuration is completed then this bit must be deasserted in order to process further AXI accesses. **NOTE:** This bit is asserted at the end of the reset sequence, meaning that the MMDC is waiting to configure and intialize the external memory before accepting any AXI accesses. Configuration request/acknowledge mechanism should be used for the following procedures: changing of timing parameters , during calibration process or driving commands via MDSCR[CMD] 0 No request to configure MMDC. 1 A request to configure MMDC is valid
14 CON_ACK	Configuration acknowledge. Whenever this bit is set, it is permitted to configure MMDC IP registers. 0 Configuration of MMDC registers is forbidden. 1 Configuration of MMDC registers is permitted.
13–11 Reserved	This read-only field is reserved and always has the value 0.
10 MRR_READ_ DATA_VALID	MRR read data valid. This field indicates that read data is valid at MDMRR register This field is relevant only for LPDDR2 mode 0 Cleared upon the assertion of MRR command 1 Set after MRR data is valid and stored at MDMRR register.
9 WL_EN	DQS pads direction. This bit controls the DQS pads direction during write-leveling calibration process. Before starting the write-leveling calibration process this bit should be set to "1". It should be set to "0" when sending write leveling exit command. For further information see Write leveling Calibration . 0 Exit write leveling mode or stay in normal mode. 1 Write leveling entry command was sent.
8–7 Reserved	This read-only field is reserved and always has the value 0.
6–4 CMD	Command. This field contains the command to be executed. This field will be automatically cleared after the command will be send to the DDR memory. 0x0 Normal operation 0x1 Precharge all, command is sent independently of bank status (set correct CMD_CS). Will be issued even if banks are closed. Mainly used for init sequence purpose.

Table continues on the next page...

MMDCx_MDSCR field descriptions (continued)

Field	Description
	0x2 Auto-Refresh Command (set correct CMD_CS). 0x3 Load Mode Register Command (DDR3, set correct CMD_CS, CMD_BA, CMD_ADDR_LSB, CMD_ADDR_MSB), MRW Command (LPDDR2, set correct CMD_CS, MR_OP, MR_ADDR) 0x4 ZQ calibration (DDR3, set correct CMD_CS, {CMD_ADDR_MSB,CMD_ADDR_LSB} = 0x400 or 0x0) 0x5 Precharge all, only if banks open (set correct CMD_CS). 0x6 MRR command (LPDDR2, set correct CMD_CS, MR_ADDR) 0x7 Reserved
3 CMD_CS	Chip Select. This field determines which chip select the command is targeted to 0 to Chip-select 0 1 to Chip-select 1
CMD_BA	Bank Address. This field determines the address of the bank within the selected chip-select where the command is targetted to. 0x0 bank address 0 0x1 bank address 1 0x2 bank address 2 0x7 bank address 7

44.12.9 MMDC Core Refresh Control Register (MMDCx_MDREF)

This register determines the refresh scheme that will be executed toward the DDR device. It specifies how often a refresh cycle occurs and how many refresh commands will be executed every refresh cycle.

For further information see Refresh Scheme .

The following tables show examples of possible refresh schemes.

Table 44-116. Refresh rate example for REF_SEL = 0

REFR[2:0]	Number of refresh commands every 64KHz	Average periodic refresh rate (tREFI)	System Refresh period
0x0	1	15.6 µs	tRFC
0x1	2	7.8 µs	2*tRFC
0x3	4	3.9µs	4*tRFC
0x7	8	1.95 µs	8*tRFC

Table 44-117. Refresh rate example for REF_SEL = 1

REFR[2:0]	Number of refresh commands every 32KHz	Average periodic refresh rate (tREFI)	System Refresh period
0x1	2	15.6 µs	2*tRFC
0x3	4	7.8 µs	4*tRFC
0x7	8	3.9µs	8*tRFC

Table 44-118. Refresh rate example for REF_SEL = 2@ 400MHz

REFR[2:0]	Number of refresh commands every refresh cycle	REF_CNT	Average periodic refresh rate (tREFI)	System Refresh period
0x0	1	0x618	3.9 µs	tRFC
0x1	2	0xC30	3.9 µs	2*tRFC
0x2	3	0x1248	3.9µs	3*tRFC
0x3	4	0x1860	3.9 µs	4*tRFC

Other refresh configurations are also allowed; the configuration values in the tables above are only examples for obtaining the desired average periodic refresh rate.

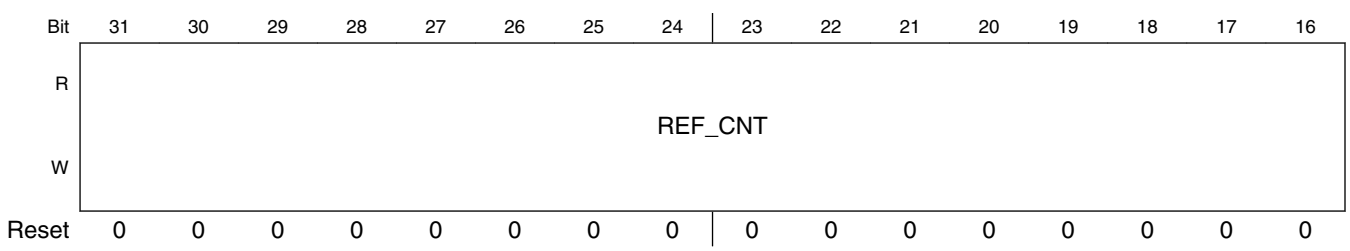
If the required average periodic refresh rate (tREFI) is kept, all of the rows will be refreshed in every refresh window. Because the memory device issues additional refresh commands for every refresh it receives, the tREFI remains the same across the device, regardless of its number of rows. This is particularly relevant in the tRFC parameter, which becomes bigger as the density increases.

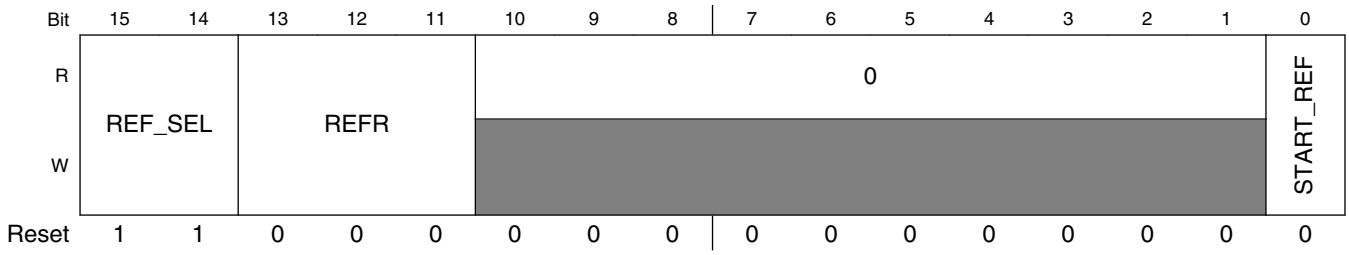
Supported Mode Of Operations:

For Channel 0: All

For Channel 1: LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 20h offset





MMDCx_MDREF field descriptions

Field	Description
31–16 REF_CNT	Refresh Counter at DDR clock period If REF_SEL equals '2' a refresh cycle will begin every amount of DDR cycles configured in this field. 0x0 Reserved. 0x1 1 cycle. 0xFFFE 65534 cycles. 0xFFFF 65535 cycles.
15–14 REF_SEL	Refresh Selector. This bit selects the source of the clock that will trigger each refresh cycle: 0 Periodic refresh cycles will be triggered in frequency of 64KHz. 1 Periodic refresh cycles will be triggered in frequency of 32KHz. 2 Periodic refresh cycles will be triggered every amount of cycles that are configured in REF_CNT field. 3 No refresh cycles will be triggered.
13–11 REFR	Refresh Rate. This field determines how many refresh commands will be issued every refresh cycle. After every refresh command the MMDC won't drive any command to the DDR device untill satisfying tRFC period 0x0 1 refresh 0x1 2 refreshes 0x2 3 refreshes 0x3 4 refreshes 0x4 5 refreshes 0x5 6 refreshes 0x6 7 refreshes 0x7 8 refreshes
10–1 Reserved	This read-only field is reserved and always has the value 0.
0 START_REF	Manual start of refresh cycle. When this field is set to '1' the MMDC will start a refresh cycle immediately according to number of refresh commands that are configured in 'REFR' field. This bit returns to zero automatically. 0 Do nothing. 1 Start a refresh cycle.

44.12.10 MMDC Core Read/Write Command Delay Register (MMDCx_MDRWD)

This register determines the delay between back to back read and write accesses. The register reset values are set to the minimum required value. As the default values are set to achieve optimal results, changing them is discouraged.

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 2Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			tDAI												
W																
Reset	0	0	0	0	1	1	1	1	1	0	0	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	RTW_SAME			WTR_DIFF			WTW_DIFF			RTW_DIFF			RTR_DIFF		
W																
Reset	0	0	0	1	0	0	1	1	0	1	1	0	1	0	0	1	0

MMDCx_MDRWD field descriptions

Field	Description
31–29 Reserved	This read-only field is reserved and always has the value 0.
28–16 tDAI	Device auto initialization period.(maximum) This field is relevant only to LPDDR2 mode 0x0 1 cycle 0xF9F 4000 cycles (Default, JEDEC value for LPDDR2, gives 10us at 400MHz clock). 0x1FFF 8192 cycles
15 Reserved	This read-only field is reserved and always has the value 0.
14–12 RTW_SAME	Read to write delay for the same chip-select. This field controls the delay between read to write commands toward the same chip select. The total delay is calculated according to: BL/2 + RTW_SAME + (tCL-tCWL) + RALAT 0x0 0 cycle 0x1 1 cycle 0x2 2 cycles (Default) 0x3 3 cycles 0x4 4 cycles 0x5 5 cycles

Table continues on the next page...

MMDCx_MDRWD field descriptions (continued)

Field	Description
	0x6 6 cycles 0x7 7 cycles
11–9 WTR_DIFF	Write to read delay for different chip-select. This field controls the delay between write to read commands toward different chip select. The total delay is calculated according to: BL/2 + WTR_DIFF + (tCL-tCWL) + RALAT 0x0 0 cycle 0x1 1 cycle 0x2 2 cycles 0x3 3 cycles (Default) 0x4 4 cycles 0x5 5 cycles 0x6 6 cycles 0x7 7 cycles
8–6 WTW_DIFF	Write to write delay for different chip-select. This field controls the delay between write to write commands toward different chip select. The total delay is calculated according to: BL/2 + WTW_DIFF 0x0 0 cycle 0x1 1 cycle 0x2 2 cycles 0x3 3 cycles (Default) 0x4 4 cycles 0x5 5 cycles 0x6 6 cycles 0x7 7 cycles
5–3 RTW_DIFF	Read to write delay for different chip-select. This field controls the delay between read to write commands toward different chip select. The total delay is calculated according to: BL/2 + RTW_DIFF + (tCL - tCWL) + RALAT 0x0 0 cycle 0x1 1 cycle 0x2 2 cycles (Default) 0x3 3 cycles 0x4 4 cycles 0x5 5 cycles 0x6 6 cycles 0x7 7 cycles
RTR_DIFF	Read to read delay for different chip-select. This field controls the delay between read to read commands toward different chip select. The total delay is calculated according to: BL/2 + RTR_DIFF 0x0 0 cycle 0x1 1 cycle 0x2 2 cycles (Default) 0x3 3 cycles 0x4 4 cycles

Table continues on the next page...

MMDCx_MDRWD field descriptions (continued)

Field	Description
	0x5 5 cycles 0x6 6 cycles 0x7 7 cycles

44.12.11 MMDC Core Out of Reset Delays Register (MMDCx_MDOR)

This register defines delays that must be kept when MMDC exits reset.

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 30h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								tXPR								0		SDE_to_RST						0		RST_to_CKE					
W																																
Reset	0	0	0	0	0	0	0	0	1	0	0	1	1	1	1	1	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1	0

MMDCx_MDOR field descriptions

Field	Description
31–24 Reserved	This read-only field is reserved and always has the value 0.
23–16 tXPR	DDR3: CKE HIGH to a valid command. This field is not relevant in LPDDR2 mode. DDR3: As defined in timing parameter table. 0x0 Reserved 0x1 2 cycles 0x2 3 cycles 0xFE 255 cycles 0xFF 256 cycles
15–14 Reserved	This read-only field is reserved and always has the value 0.
13–8 SDE_to_RST	DDR3: Time from SDE enable until DDR reset# is high. In LPDDR2 mode this field is not relevant . **NOTE:** Each cycle in this field is 15.258 us. 0x0 Reserved 0x1 Reserved

Table continues on the next page...

MMDCx_MDOR field descriptions (continued)

Field	Description
	0x2 Reserved 0x3 1 cycles 0x4 2 cycles 0x10 14 cycles (Jedec value for DDR3) - total of 200 us 0x3E 60 cycles 0x3F 61 cycles
7–6 Reserved	This read-only field is reserved and always has the value 0.
RST_to_CKE	DDR3: Time from SDE enable to CKE rise. In case that DDR reset# is low, will wait until it's high and thenwait this period until rising CKE. (JEDEC value is 500 us) LPDDR2: Idle time ater first CKE assertion. (JEDEC value is 200 us) **NOTE:** Each cycle in this field is 15.258 us. 0x0 Reserved 0x1 Reserved 0x2 Reserved 0x3 1 cycles 0x10 14 cycles (JEDEC value for LPDDR2) - total of 200 us 0x23 33 cycles (JEDEC value for DDR3) - total of 500 us 0x3E 60 cycles 0x3F 61 cycles

44.12.12 MMDC Core MRR Data Register (MMDCx_MDMRR)

This register contains data that was collected after issuing MRR command. The data in this register is valid only when MDSCR[MRR_READ_DATA_VALID] is set to "1".

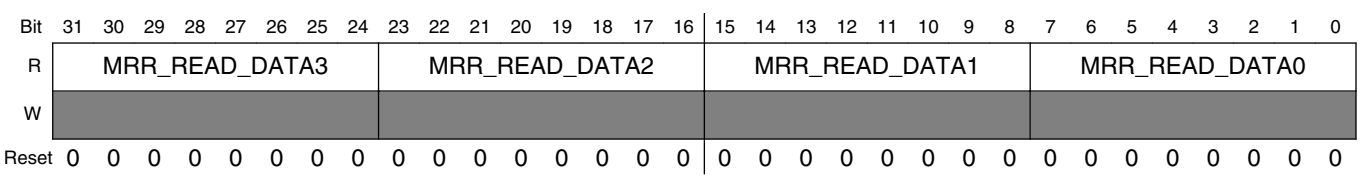
This register is relevant only in LPDDR2 mode. For further information see LPDDR2 Refresh Rate Update and Timing Derating .

Supported Mode Of Operations:

For Channel 0: LP2_2ch_x16, LP2_2ch_x32

For Channel 1: LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 34h offset



MMDCx_MDMRR field descriptions

Field	Description
31–24 MRR_READ_DATA3	MRR DATA that arrived on DQ[31:24]
23–16 MRR_READ_DATA2	MRR DATA that arrived on DQ[23:16]
15–8 MRR_READ_DATA1	MRR DATA that arrived on DQ[15:8]
MRR_READ_DATA0	MRR DATA that arrived on DQ[7:0]

44.12.13 MMDC Core Timing Configuration Register 3 (MMDCx_MDCFG3LP)

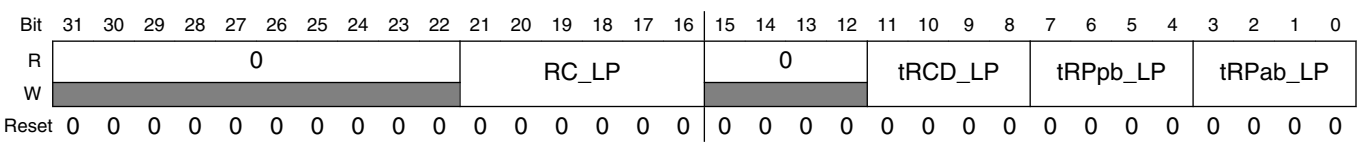
This register is relevant only for LPDDR2 mode.

Supported Mode Of Operations:

For Channel 0: LP2_2ch_x16, LP2_2ch_x32

For Channel 1: LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 38h offset



MMDCx_MDCFG3LP field descriptions

Field	Description
31–22 Reserved	This read-only field is reserved and always has the value 0.
21–16 RC_LP	Active to Active or Refresh command period (same bank). (This field is valid only for LPDDR2 memories) 0x0 1 clock 0x1 2 clocks 0x2 3 clocks 0x3E 63 clocks 0x3F Reserved
15–12 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

MMDCx_MDCFG3LP field descriptions (continued)

Field	Description
11–8 tRCD_LP	Active command to internal read or write delay time (same bank). (This field is valid only for LPDDR2 memories) 0x0 1 clock 0x1 2 clocks 0x2 3 clocks 0xE 15 clocks 0xF Reserved
7–4 tRPpb_LP	Precharge (per bank) command period (same bank). (This field is valid only for LPDDR2 memories) 0x0 1 clock 0x1 2 clocks 0x2 3 clocks 0xE 15 clocks 0xF Reserved
tRPab_LP	Precharge (all banks) command period. (This field is valid only for LPDDR2 memories) 0x0 1 clock 0x1 2 clocks 0x2 3 clocks 0xE 15 clocks 0xF Reserved

44.12.14 MMDC Core MR4 Derating Register (MMDCx_MDMR4)

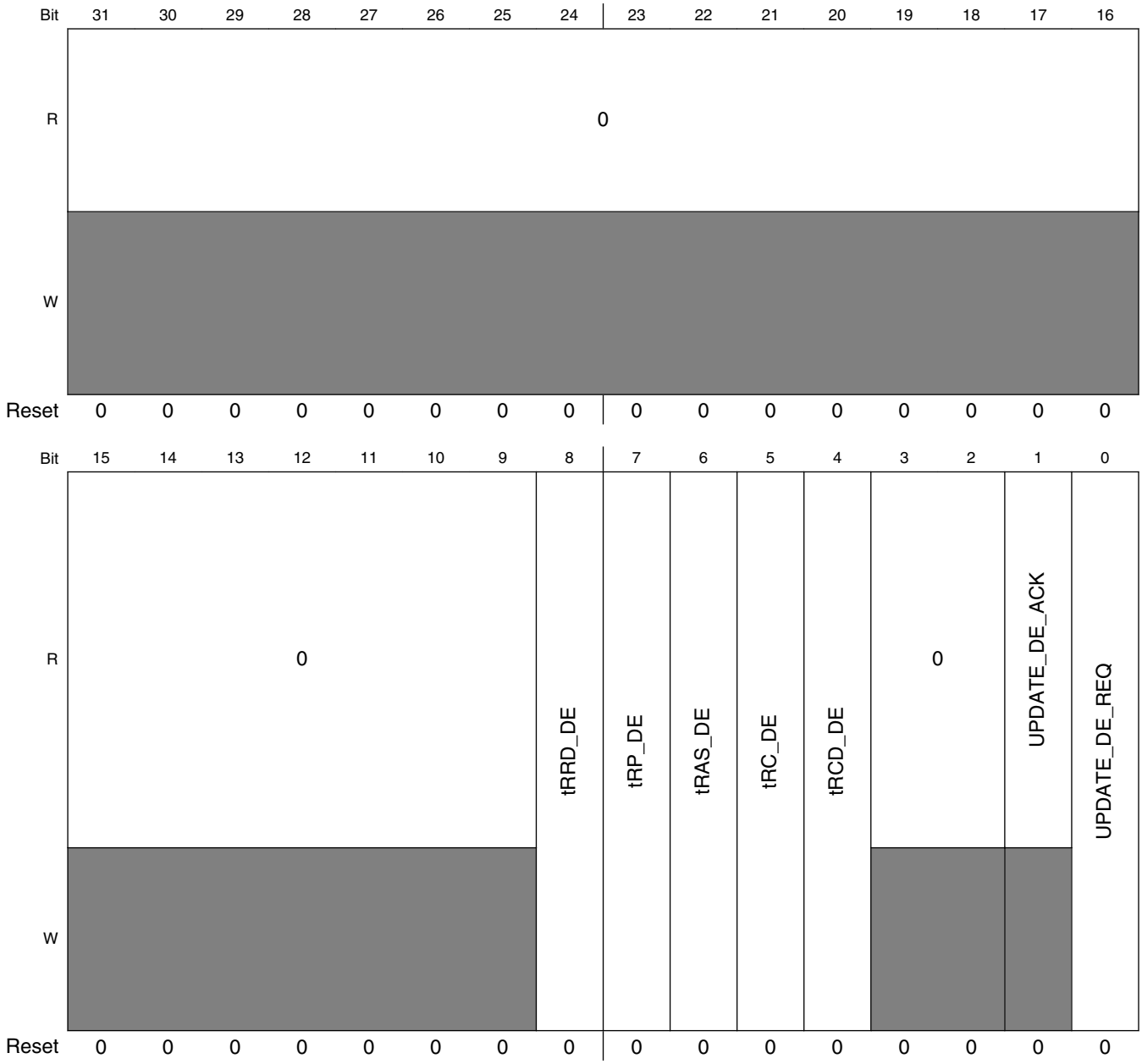
This register is relevant only for LPDDR2 mode. It is used to dynamically change certain values depending on MR4 read result, which is based on memory temperature sensor result.

Supported Mode of Operations:

For Channel 0: LP2_2ch_x16, LP2_2ch_x32

For Channel 1: LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 3Ch offset



MMDCx_MDMR4 field descriptions

Field	Description
31–9 Reserved	This read-only field is reserved and always has the value 0.
8 tRRD_DE	tRRD derating value. 0 Original tRRD is used. 1 tRRD is derated in 1 cycle.
7 tRP_DE	tRP derating value.

Table continues on the next page...

MMDCx_MDMR4 field descriptions (continued)

Field	Description
	0 Original tRP is used. 1 tRP is derated in 1 cycle.
6 tRAS_DE	tRAS derating value. 0 Original tRAS is used. 1 tRAS is derated in 1 cycle.
5 tRC_DE	tRC derating value. 0 Original tRC is used. 1 tRC is derated in 1 cycle.
4 tRCD_DE	tRCD derating value. 0 Original tRCD is used. 1 tRCD is derated in 1 cycle.
3–2 Reserved	This read-only field is reserved and always has the value 0.
1 UPDATE_DE_ACK	Update Derated Values Acknowledge. This read only bit will be cleared upon UPDATE_DE_REQ assertion and will be set after the new values are taken.
0 UPDATE_DE_REQ	Update Derated Values Request. This read modify write field is automatically cleared after the request is issued. 0 Do nothing. 1 Request to update the following values: tRRD, tRCD, tRP, tRC, tRAS and refresh related fields(MDREF register): REF_CNT, REF_SEL, REFR

44.12.15 MMDC Core Address Space Partition Register (MMDCx_MDASP)

This register defines the partitioning between chip select 0 and chip select 1. For further information see Chip select settings .

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 40h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																CS0_END															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

MMDCx_MDASP field descriptions

Field	Description
31–7 Reserved	This read-only field is reserved and always has the value 0.
CS0_END	CS0_END. Defines the absolute last address associated with CS0 with increments of 256Mb. CS0_END=AXI_ADDRESS[31:25] bits. In DDR3 and 1-channel LPDDR2 mode: MMDCx_MDASP[CS0_END] should be set to DDR_CS_SIZE/32MB + 0x7 (DDR base address begins at 0x10000000) In 2-channel LPDDR2 mode: MMDC0_MDASP[CS0_END] should be set to DDR_CS_SIZE/32M + 0x3f (channel 0 base address begins at 0x80000000) MMDC1_MDASP[CS0_END] should be set to DDR_CS_SIZE/32M + 0x7 (channel 1 base address begins at 0x10000000) In 2-channel LPDDR2 with 4k-interleave mode: MMDC0_MDASP[CS0_END] should be set to DDR_CS_SIZE/32M + 0x43 MMDC1_MDASP[CS0_END] should be set to DDR_CS_SIZE/32M + 0x3

44.12.16 MMDC Core AXI Reordering Control Regsiter (MMDCx_MAARCR)

This register determines the values of the weights used for the re-ordering arbitration engine. For further information see Performance .

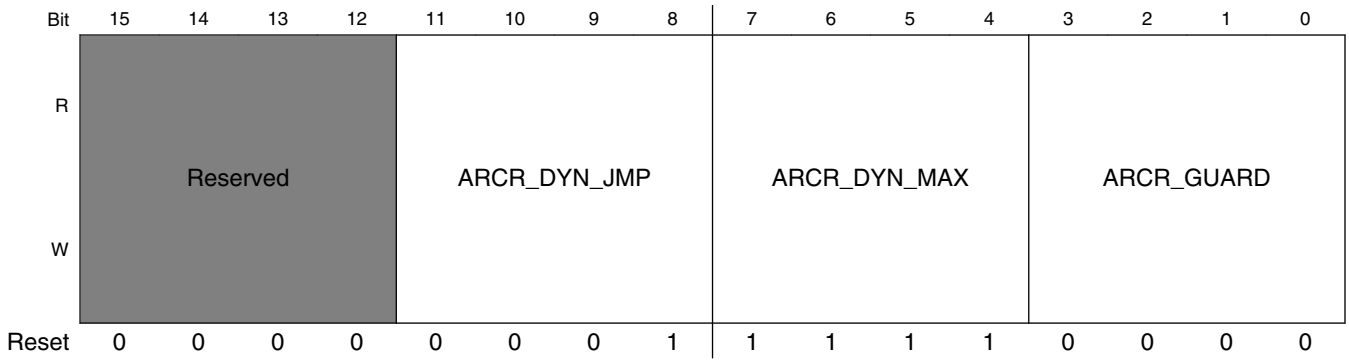
Supported Mode Of Operations:

For Channel 0: All

For Channel 1: LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 400h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ARCR_SEC_ERR_LOCK	ARCR_SEC_ERR_EN	Reserved	ARCR_EXC_ERR_EN	Reserved			ARCR_RCH_EN	Reserved	ARCR_PAG_HIT			Reserved	ARCR_ACC_HIT		
Reset	0	1	0	1	0	0	0	1	0	1	0	0	0	0	1	0



MMDCx_MAARCR field descriptions

Field	Description
31 ARCR_SEC_ERR_LOCK	Once set, this bit locks ARCR_SEC_ERR_EN and prevents from its updating. This bit can be only cleared by reset Default value is 0x0 - encoding 0 (unlocked) 0 ARCR_SEC_ERR_EN is unlocked, so can be updated any moment 1 ARCR_SEC_ERR_EN is locked, so it can't be updated
30 ARCR_SEC_ERR_EN	This bit defines whether security read/write access violation result in SLV Error response or in OKAY response Default value is 0x1 - encoding 1(response is SLV Error, rresp/bresp=2'b10) 0 security violation results in OKAY response (rresp/bresp=2'b00) 1 security violation results in SLAVE Error response (rresp/bresp=2'b10)
29 -	This field is reserved. Reserved
28 ARCR_EXC_ERR_EN	This bit defines whether exclusive read/write access violation of AXI 6.2.4 rule result in SLV Error response or in OKAY response Default value is 0x1 - encoding 1(response is SLV Error) 0 violation of AXI exclusive rules (6.2.4) result in OKAY response (rresp/bresp=2'b00) 1 violation of AXI exclusive rules (6.2.4) result in SLAVE Error response (rresp/bresp=2'b10)
27–25 -	This field is reserved. Reserved
24 ARCR_RCH_EN	This bit defines whether Real time channel is activated and bypassed all other pending accesses, So accesses with QoS=='F' will be granted the highest prioritiy in the optimization/reordering mechanism Default value is 0x1 - encoding 1 (Enabled) 0 normal prioritization, no bypassing 1 accesses with QoS=='F' bypass the arbitration
23 -	This field is reserved. Reserved
22–20 ARCR_PAG_HIT	ARCR Page Hit Rate. This value will be added by the optimization/reordering mechanism to any pending access that is targeted to an open DDR row. Default value of ARCR_PAG_HIT is 0x00100 - encoding 4.
19 -	This field is reserved. Reserved

Table continues on the next page...

MMDCx_MAARCR field descriptions (continued)

Field	Description
18–16 ARCR_ACC_HIT	ARCR Access Hit Rate. This value will be added by the optimization/reordering mechanism to any pending access that has the same access type (read/write) as the previous access. Default value of is ARCR_ACC_HIT 0x0010 - encoding 2.
15–12 -	This field is reserved. Reserved
11–8 ARCR_DYN_JMP	ARCR Dynamic Jump. Each time an access wan't chosen by the optimization/reordering mechanism then its dynamic score will be incremented by ARCR_DYN_JMP value. **NOTE:** Setting ARCR_DYN_JMP may cause starvation of low priority accesses **NOTE:** ARCR_DYN_JMP must be smaller than ARCR_DYN_MAX Default ARCR_DYN_JMP value is 0x0001 - encoding 1
7–4 ARCR_DYN_MAX	ARCR Dynamic Maximum. ARCR_DYN_MAX is the maximum dynamic score value that each access inside the optimization/reordering mechanism can get. 0000 0 0001 1 1111 15 (default)
ARCR_GUARD	ARCR Guard. After an access reached the maximum dynamic score value, it will wait additional ARCR_GUARD arbitration cycles and then will gain the highest priority in the optimization/reordering mechanism. 0000 15 (default) 0001 16 1111 30

44.12.17 MMDC Core Power Saving Control and Status Register (MMDCx_MAPSR)

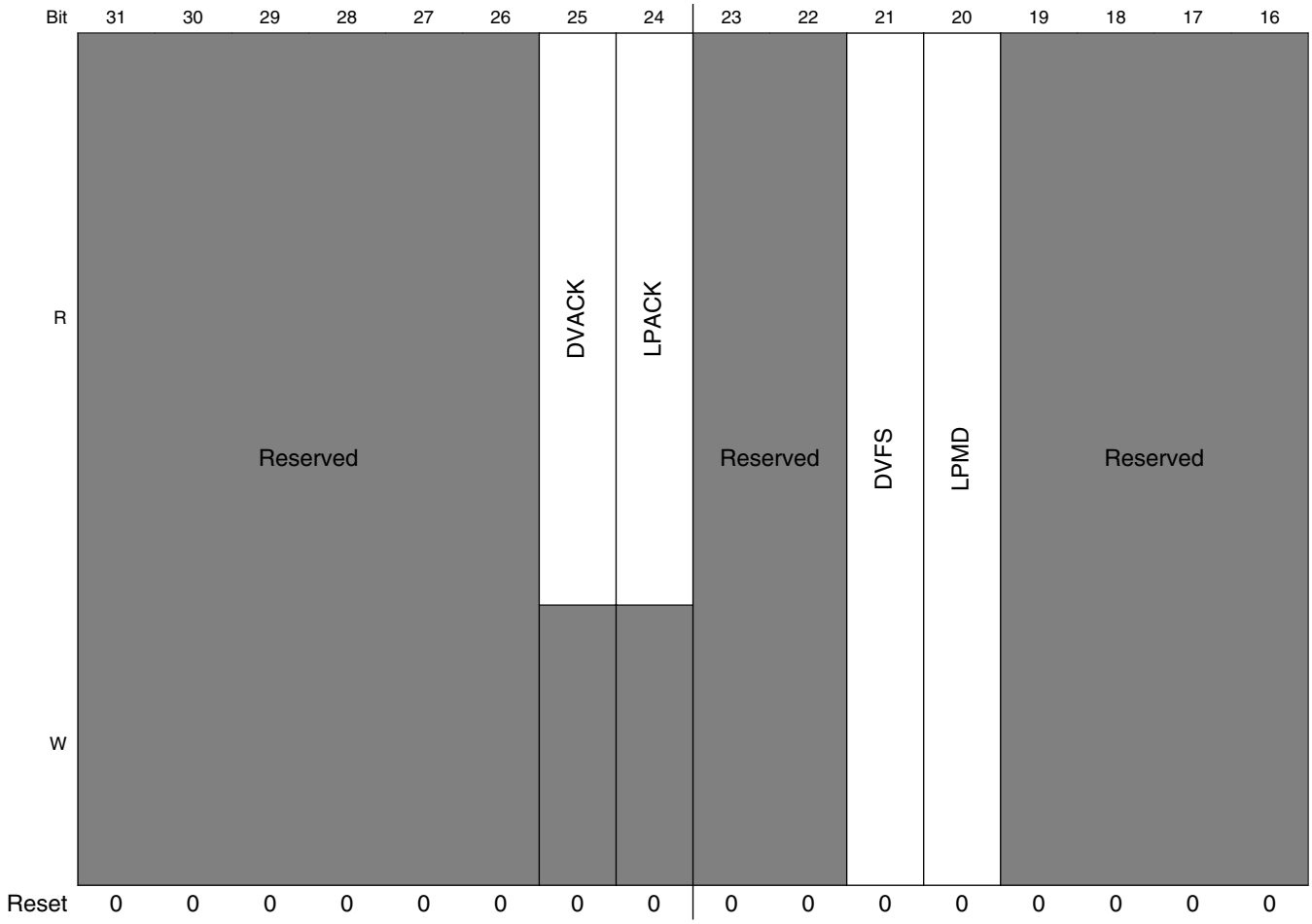
The MAPSR determines the power saving features of MMDC. For further information see Power Saving and Clock Frequency Change modes .

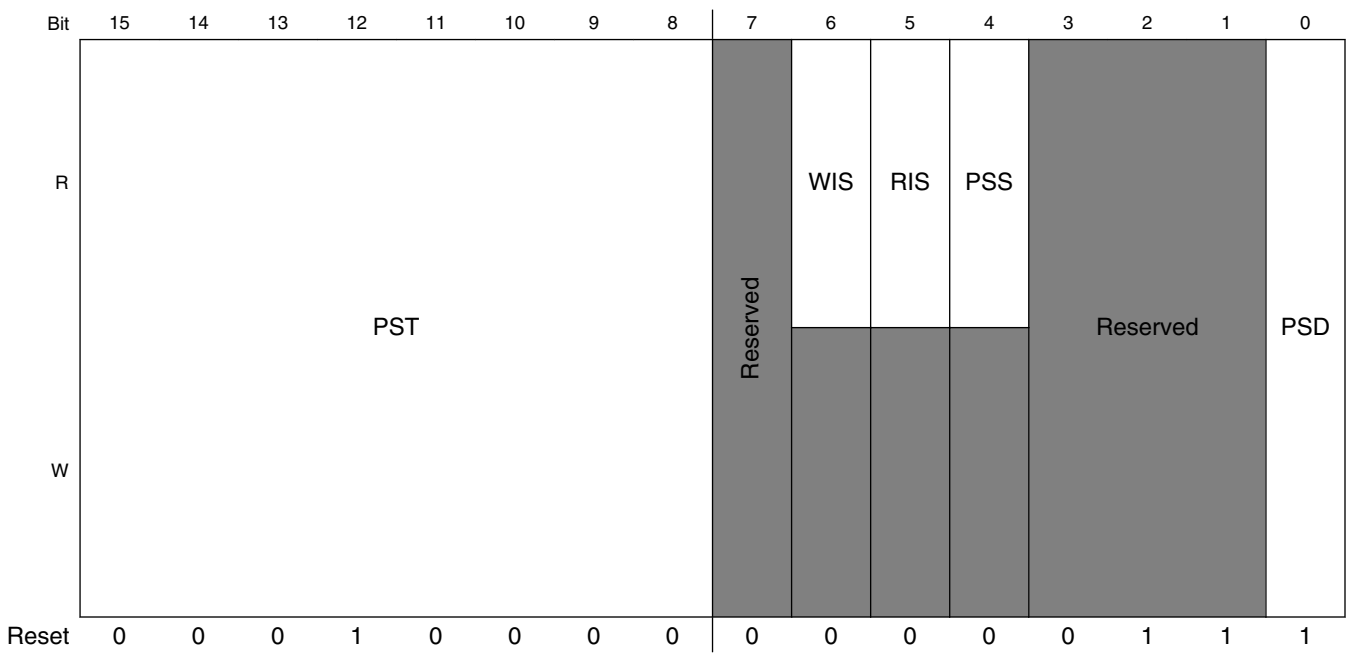
Supported Mode Of Operations:

For Channel 0: All

For Channel 1: LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 404h offset





MMDCx_MAPSR field descriptions

Field	Description
31–26 -	This field is reserved. Reserved
25 DVACK	General DVFS acknowledge. This read only bit indicates whether a dvfs acknowledge was asserted and that MMDC is in self-refresh mode
24 LPACK	General low-power acknowledge. This read only bit indicates whether a low-power acknowledge was asserted and that MMDC is in self-refresh mode
23–22 -	This field is reserved. Reserved
21 DVFS	General DVFS request. SW request for DVFS. Assertion of this bit will yield in self-refresh entry sequence 0 no dvfs request 1 dvfs request
20 LPMD	General LPMD request. SW request for LPMD. Assertion of this bit will yield in self-refresh entry sequence 0 no lpmd request 1 lpmd request
19–16 -	This field is reserved. Reserved
15–8 PST	Automatic Power saving timer. Valid only when PSD is set to "0". When the MMDC is idle for amount of cycles specified in that field then the DDR device will be entered automatically into self-refresh mode. The real value which is used is register-value multiplied by 64. 00000000 Reserved - this value is forbidden. 00000001 timer is configured to 64 clock cycles. 00000010 timer is configured to 128 clock cycles. 00010000 (Default)- 1024 clock cycles. 11111111 timer clock is configured to 16320 clock cycles.
7 -	This field is reserved. Reserved.
6 WIS	Write Idle Status.This read only bit indicates whether write request buffer is idle (empty) or not. 0 idle 1 not idle
5 RIS	Read Idle Status.This read only bit indicates whether read request buffer is idle (empty) or not. 0 idle 1 not idle
4 PSS	Power Saving Status. This read only bit indicates whether the MMDC is in automatic power saving mode. 0 not in power saving 1 power saving
3–1 -	This field is reserved. Reserved.
0 PSD	Automatic Power Saving Disable. When the value of PSD is "0" (i.e automatic power saving is enabled) then the PST is activated and MMDC will enter automatically to self-refresh while the number of idle cycle reached.

Table continues on the next page...

MMDCx_MAPSR field descriptions (continued)

Field	Description
	NOTE: This bit must be disabled (i.e set to "1") during calibration process 0 power saving enabled 1 power saving disabled (default)

44.12.18 MMDC Core Exclusive ID Monitor Register0 (MMDCx_MAEXIDR0)

This register defines the ID to be monitored for exclusive accesses of monitor0 and monitor1. For further information see Exclusive accesses handling .

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 408h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	EXC_ID_MONITOR1																EXC_ID_MONITOR0															
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDCx_MAEXIDR0 field descriptions

Field	Description
31–16 EXC_ID_ MONITOR1	This feild defines ID for Exclusive monitor#1. Default value is 0x0020
EXC_ID_ MONITOR0	This feild defines ID for Exclusive monitor#0. Default value is 0x0000

44.12.19 MMDC Core Exclusive ID Monitor Register1 (MMDCx_MAEXIDR1)

This register defines the ID to be monitored for exclusive accesses of monitor2 and monitor3. For further information see Exclusive accesses handling .

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 40Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	EXC_ID_MONITOR3																EXC_ID_MONITOR2															
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

MMDCx_MAEXIDR1 field descriptions

Field	Description
31–16 EXC_ID_ MONITOR3	This feild defines ID for Exclusive monitor#3. Default value is 0x0060
EXC_ID_ MONITOR2	This feild defines ID for Exclusive monitor#2. Default value is 0x0040

44.12.20 MMDC Core Debug and Profiling Control Register 0 (MMDCx_MADPCR0)

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 410h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0						SBS	SBS_EN	0				CYC_OVF	PRF_FRZ	DBG_RST	DBG_EN
W													w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDCx_MADPCR0 field descriptions

Field	Description
31–10 Reserved	This read-only field is reserved and always has the value 0.
9 SBS	Step By Step trigger. If SBS_EN is set to "1" then dispatching AXI pending access toward the DDR will done only if this bit is set to "1", otherwise no access will be dispatched toward the DDR. This bit is cleared when the pending access has been issued toward the DDR device. 1 Launch AXI pending access toward the DDR 0 No access will be launched toward the DDR
8 SBS_EN	Step By Step debug Enable. Enable step by step mode. Every time this mechanism is enabled then setting SBS to "1" will dispatch one pending AXI access to the DDR and in parallel its attributes will be observed in the status registes (MASBS0 and MASBS1). For further information see Step By Step (SBS) software monitor . 0 disable 1 enable
7–4 Reserved	This read-only field is reserved and always has the value 0.
3 CYC_OVF	Total Profiling Cycles Count Overflow. When profiling mechanism is enabled (DBG_EN is set to "1") then this bit is asserted when overflow of CYC_COUNT occurred. Cleared by writing 1 to it. 0 no overflow 1 overflow
2 PRF_FRZ	Profiling freeze. When this bit is assertted then the profilling mechanism will be freezed and the associated status registers (MADPSR0-MADPSR5) will hold the the current profiling values. 0 profiling counters are not frozen 1 profiling counters are frozen
1 DBG_RST	Debug and Profiling Reset. Reset all debug and profiling counters and components. 0 no reset 1 reset
0 DBG_EN	Debug and Profiling Enable. Enable debug and profiling mechanism. When this bit is asserted then the MMDC will perform a profiling based on the ID that is configured to MADPCR1. Upon assertion of PRF_FRZ the profiling will be freezed and the profiling results will be sampled to the status registers (MADPSR0-MADPSR5). For further information see MMDC Profiling . default is "disable" 0 disable 1 enable

44.12.21 MMDC Core Debug and Profiling Control Register 1 (MMDCx_MADPCR1)

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 414h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PRF_AXI_ID_MASK																PRF_AXI_ID															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDCx_MADPCR1 field descriptions

Field	Description
31–16 PRF_AXI_ID_ MASK	Profiling AXI ID Mask. AXI ID bits which masked by this value are chosen for profiling. 1 AXI ID specific bit is chosen for profiling 0 AXI ID specific bit is ignored (don't care)
PRF_AXI_ID	Profiling AXI ID. AXI IDs that matches a bit-wise AND logic operation between PRF_AXI_ID and PRF_AXI_ID_MASK are chosen for profiling. Default value is 0x0, to choose any ID-s for profiling

44.12.22 MMDC Core Debug and Profiling Status Register 0 (MMDCx_MADPSR0)

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 418h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CYC_COUNT																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDCx_MADPSR0 field descriptions

Field	Description
CYC_COUNT	Total Profiling cycle Count. This field reflects the total cycle count in case the profiling mechanism is enabled from assertion of DBG_EN and until PRF_FRZ is asserted

44.12.23 MMDC Core Debug and Profiling Status Register 1 (MMDCx_MADPSR1)

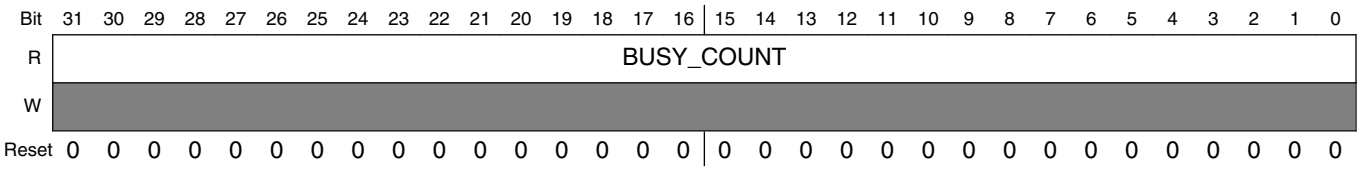
The register reflects the total cycles during which the MMDC state machines were busy (both writes and reads). This information can be used for DDR Utilization calculation.

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 41Ch offset



MMDCx_MADPSR1 field descriptions

Field	Description
BUSY_COUNT	Profiling Busy Cycles Count. This field reflects the total number of cycles where the MMDC read and write state machines were busy during the profiling period. Can be used for DDR utilization calculations. Busy cycles are any MMDC clock cycles where the internal state machine is not idle. If any read or write requests are pending in the FIFOs, the MMDC is not idle.

44.12.24 MMDC Core Debug and Profiling Status Register 2 (MMDCx_MADPSR2)

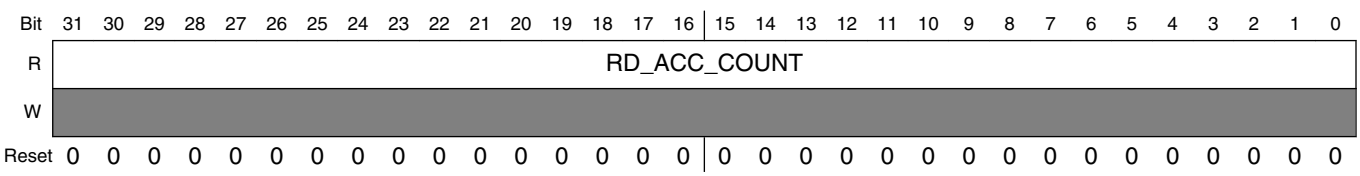
This register reflects the total number of read accesses (per AXI ID) toward MMDC.

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 420h offset



MMDCx_MADPSR2 field descriptions

Field	Description
RD_ACC_ COUNT	Profiling Read Access Count. This register reflects the total number of read accesses (per AXI ID) toward MMDC.

44.12.25 MMDC Core Debug and Profiling Status Register 3 (MMDCx_MADPSR3)

This register reflects the total number of write accesses (per AXI ID) toward MMDC.

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 424h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WR_ACC_COUNT																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDCx_MADPSR3 field descriptions

Field	Description
WR_ACC_ COUNT	Profiling Write Access Count. This register reflects the total number of write accesses (per AXI ID) toward MMDC.

44.12.26 MMDC Core Debug and Profiling Status Register 4 (MMDCx_MADPSR4)

This register reflects the total number of bytes that were transferred during read access (per AXI ID) toward MMDC.

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 428h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RD_BYTES_COUNT																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDCx_MADPSR4 field descriptions

Field	Description
RD_BYTES_ COUNT	Profiling Read Bytes Count. This register reflects the total number of bytes that were transferred during read access (per AXI ID) toward MMDC.

44.12.27 MMDC Core Debug and Profiling Status Register 5 (MMDCx_MADPSR5)

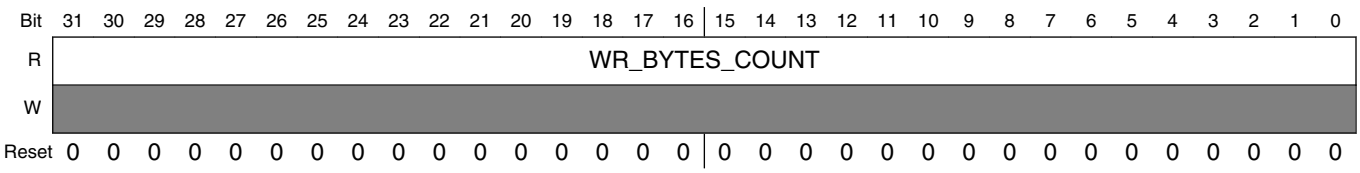
This register reflects the total number of bytes that were transferred during write access (per AXI ID) toward MMDC.

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 42Ch offset



MMDCx_MADPSR5 field descriptions

Field	Description
WR_BYTES_ COUNT	Profiling Write Bytes Count. This register reflects the total number of bytes that were transferred during write access (per AXI ID) toward MMDC.

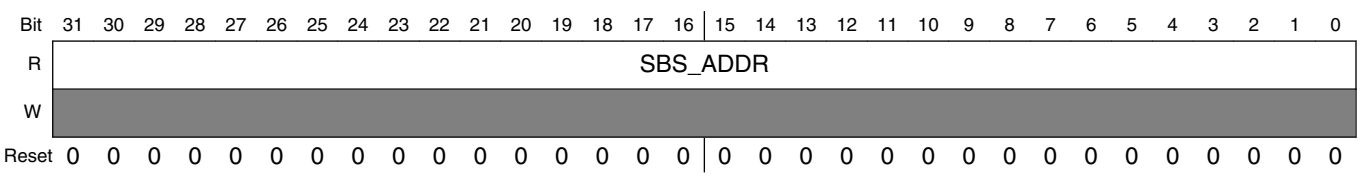
44.12.28 MMDC Core Step By Step Address Register (MMDCx_MASBS0)

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 430h offset



MMDCx_MASBS0 field descriptions

Field	Description
SBS_ADDR	Step By Step Address. These bits reflect the address of the pending request in case of step by step mode.

44.12.29 MMDC Core Step By Step Address Attributes Register (MMDCx_MASBS1)

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 434h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SBS_AXI_ID															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SBS_LEN			SBS_BUFF	SBS_BURST		SBS_SIZE		SBS_PROT				SBS_LOCK		SBS_TYPE	SBS_VLD
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDCx_MASBS1 field descriptions

Field	Description
31–16 SBS_AXI_ID	Step By Step AXI ID. These bits reflect the AXI ID of the pending request in case of step by step mode.
15–13 SBS_LEN	Step By Step Length. These bits reflect the AXI LENGTH of the pending request in case of step by step mode. 000 burst of length 1 001 burst of length 2 111 burst of length 8
12 SBS_BUFF	Step By Step Buffered. This bit reflect the AXI CACHE[0] of the pending request in case of step by step mode. Relevant only for write requests
11–10 SBS_BURST	Step By Step Burst. These bits reflect the AXI BURST of the pending request in case of step by step mode. 00 FIXED 01 INCR burst 10 WRAP burst 11 reserved
9–7 SBS_SIZE	Step By Step Size. These bits reflect the AXI SIZE of the pending request in case of step by step mode. 000 8 bits 001 16 bits 010 32 bits 011 64 bits

Table continues on the next page...

MMDCx_MASBS1 field descriptions (continued)

Field	Description
	100 128bits 101-111 Reserved
6–4 SBS_PROT	Step By Step Protection. These bits reflect the AXI PROT of the pending request in case of step by step mode.
3–2 SBS_LOCK	Step By Step Lock. These bits reflect the AXI LOCK of the pending request in case of step by step mode.
1 SBS_TYPE	Step By Step Request Type. These bits reflect the type (read/write) of the pending request in case of step by step mode. 0 write 1 read
0 SBS_VLD	Step By Step Valid. This bit reflects whether there is a pending request in case of step by step mode. 0 not valid 1 valid

44.12.30 MMDC Core General Purpose Register (MMDCx_MAGENP)

This register is a general 32 bit read/write register.

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 440h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GP31_GP0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDCx_MAGENP field descriptions

Field	Description
GP31_GP0	General purpose read/write bits.

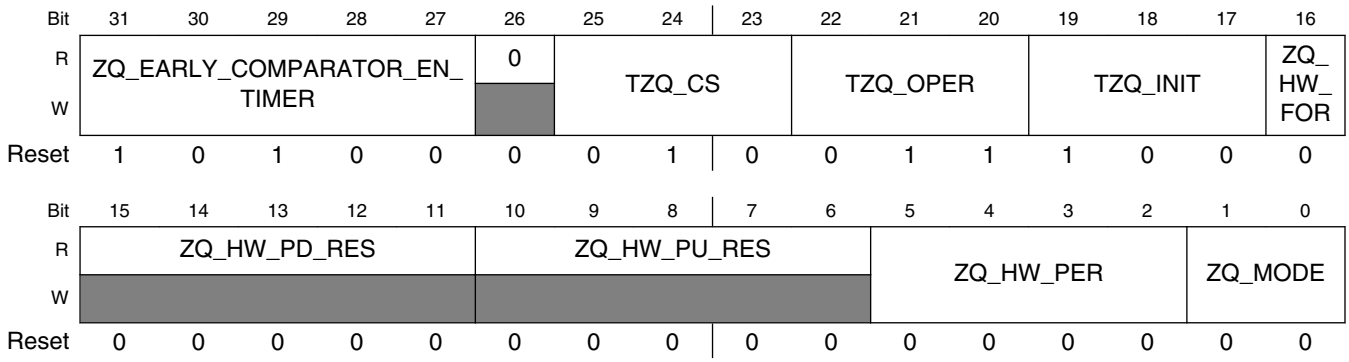
44.12.31 MMDC PHY ZQ HW control register (MMDCx_MPZQHWCTRL)

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: This register is reserved for channel 1. Channel 1 ZQ is also controlled by MMDC0_MPZQHWCTRL.

Address: Base address + 800h offset



MMDCx_MPZQHWCTRL field descriptions

Field	Description
31–27 ZQ_EARLY_COMPARATOR_EN_TIMER	ZQ early comparator enable timer. This timer defines the interval between the warming up of the comparator of the i.MX ZQ calibration pad and the begining of the ZQ calibration process with the pad 0x0 - 0x6 Reserved 0x7 8 cycles 0x14 21 cycles (Default) 0x1E 31 cycles 0x1F 32 cycles
26 Reserved	This read-only field is reserved and always has the value 0.
25–23 TZQ_CS	Device ZQ short time. This field holds the number of cycles that are required by the external DDR device to perform ZQ short calibration. Upon driving the command to the DDR device then no further accesses will be issued to the DDR device till satisfying that time. **NOTE:** In LPDDR2 the ZQ short time is taken from MPZQLP2CTL[ZQ_LP2_HW_ZQCS] **NOTE:** This field should not be update during ZQ calibration. 000 Reserved 001 Reserved 010 128 cycles (Default) 011 256 cycles 100 512 cycles 101 1024 cycles 110- 111 Resreved
22–20 TZQ_OPER	Device ZQ long/oper time. This field holds the number of cycles that are required by the external DDR device to perform ZQ long calibration except the first ZQ long command that is isued after reset. Upon driving the command to the DDR device then no further accesses will be issued to the DDR device till satisfying that time. **NOTE:** In LPDDR2 the ZQ oper time is taken from MPZQLP2CTL[ZQ_LP2_HW_ZQCL] **NOTE:** This field should not be update during ZQ calibration. 000 Reserved

Table continues on the next page...

MMDCx_MPZQHWCTRL field descriptions (continued)

Field	Description
	001 Reserved 010 128 cycles 011 256 cycles - Default (JEDEC value for DDR3) 100 512 cycles 101 1024 cycles 110- 111 Resreved
19–17 TZQ_INIT	Device ZQ long/init time. This field holds the number of cycles that are required by the external DDR device to perform ZQ long calibration right after reset. Upon driving the command to the DDR device then no further accesses will be issued to the DDR device till satisfying that time. **NOTE:** In LPDDR2 the ZQ init time is taken from MPZQLP2CTL[ZQ_LP2_HW_ZQINIT] **NOTE:** This field should not be update during ZQ calibration. 000 Reserved 001 Reserved 010 128 cycles 011 256 cycles 100 512 cycles - Default (JEDEC value for DDR3) 101 1024 cycles 110- 111 Resreved
16 ZQ_HW_FOR	Force ZQ automatic calibration process with the i.MX ZQ calibration pad. When this bit is asserted then the MMDC will issue one ZQ automatic calibration process with the i.MX ZQ calibration pad. It is the user responsibility to make sure that all the accesses to DDR will be finished before asserting this bit using CON_REQ/CON_ACK mechanism. HW will negate this bit upon completion of the ZQ calibration process. Upon negation of this bit the ZQ HW calibration pull-up and pull-down results (ZQ_HW_PU_RES and ZQ_HW_PD_RES respectively) are valid **NOTE:** In order to enable this bit ZQ_MODE must be set to either "1" or "3"
15–11 ZQ_HW_PD_RES	ZQ HW calibration pull-down result. This field holds the pull-down resistor value calculated at the end of the ZQ automatic calibration process with the i.MX ZQ calibration pad. 00000 Max. resistance. 11111 Min. resistance.
10–6 ZQ_HW_PU_RES	ZQ automatic calibration pull-up result. This field holds the pull-up resistor value calculated at the end of the ZQ automatic calibration process with the i.MX ZQ calibration pad. 00000 Min. resistance. 11111 Max. resistance.
5–2 ZQ_HW_PER	ZQ periodic calibration time. This field determines how often the periodic ZQ calibration is performed. This field is applied for both ZQ short calibration and ZQ automatic calibration process with i.MX ZQ calibration pad. Whenever this timer is expired then according to ZQ_MODE the ZQ automatic calibration process with the i.MX ZQ calibration pad will be issued and/or short/long command will be issued to the external DDR device. This field is ignored if ZQ_MODE equals "00" 0000 ZQ calibration is performed every 1 ms. 0001 ZQ calibration is performed every 2 ms. 0010 ZQ calibration is performed every 4 ms. 1010 ZQ calibration is performed every 1 sec.

Table continues on the next page...

MMDCx_MPZQHWCTRL field descriptions (continued)

Field	Description
	1110 ZQ calibration is performed every 16 sec. 1111 ZQ calibration is performed every 32 sec.
ZQ_MODE	ZQ calibration mode: 0x0 No ZQ calibration is issued. (Default) 0x1 ZQ calibration is issued to i.MX ZQ calibration pad together with ZQ long command to the external DDR device only when exiting self refresh. 0x2 ZQ calibration command long/short is issued only to the external DDR device periodically and when exiting self refresh 0x3 ZQ calibration is issued to i.MX ZQ calibration pad together with ZQ calibration command long/ short to the external DDR device periodically and when exiting self refresh

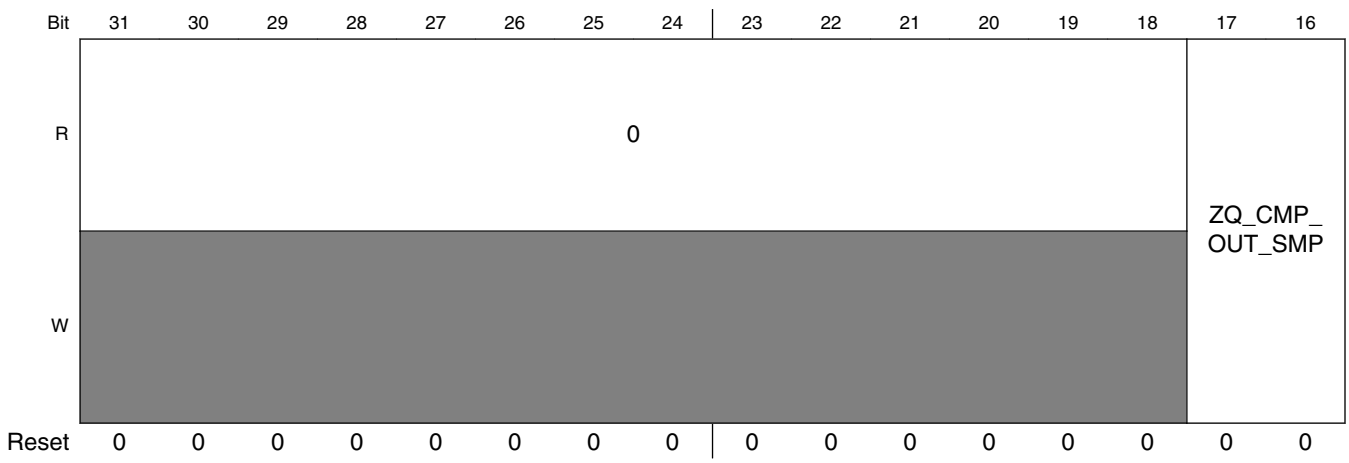
44.12.32 MMDC PHY ZQ SW control register (MMDCx_MPZQSWCTRL)

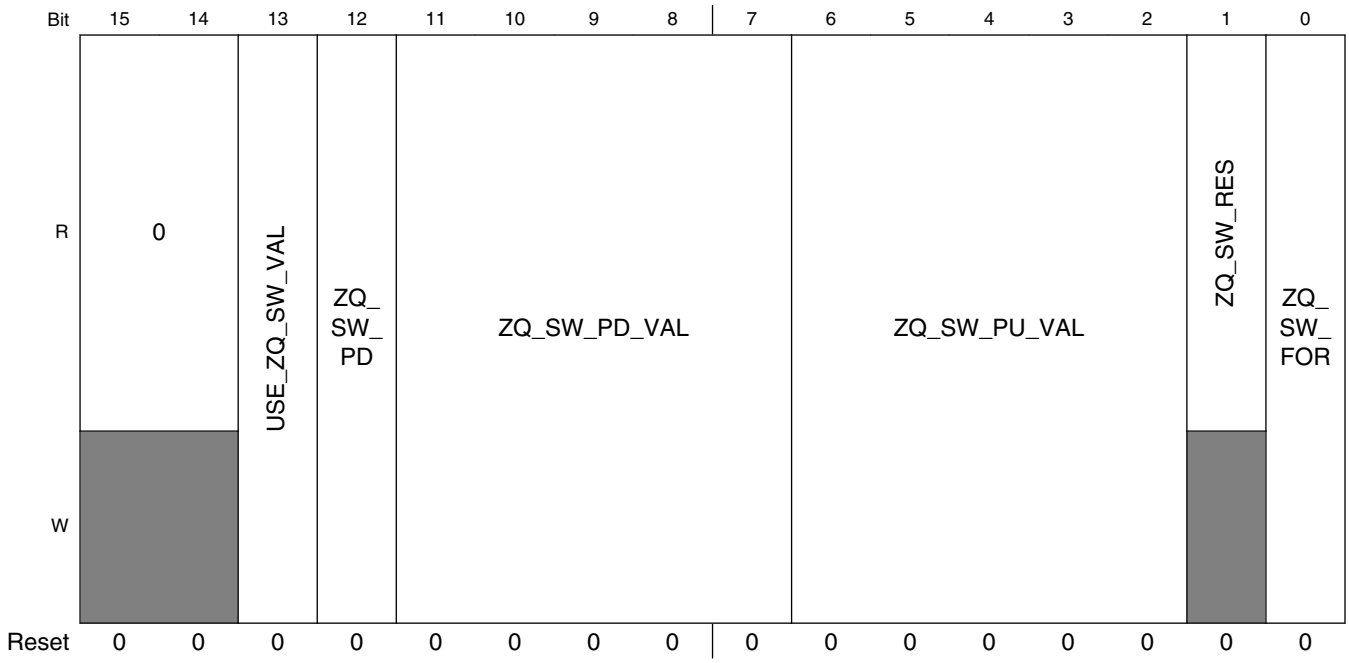
Supported Mode Of Operations:

For Channel 0: All

For Channel 1: This register is reserved.

Address: Base address + 804h offset





MMDCx_MPZQSWCTRL field descriptions

Field	Description
31–18 Reserved	This read-only field is reserved and always has the value 0.
17–16 ZQ_CMP_OUT_SMP	Defines the amount of cycles between driving the ZQ signals to the ZQ pad and till sampling the comparator enable output while performing ZQ calibration process with the i.MX ZQ calibration pad 00 7 cycles 01 15 cycles 10 23 cycles 11 31 cycles
15–14 Reserved	This read-only field is reserved and always has the value 0.
13 USE_ZQ_SW_VAL	Use SW ZQ configured value for I/O pads resistor controls. This bit selects whether ZQ SW value or ZQ HW value will be driven to the I/O pads resistor controls. By default this bit is cleared and MMDC drives the HW ZQ status bits on the resistor controls of the I/O pads. **NOTE:** This bit should not be updated during ZQ calibration. 0 Fields ZQ_HW_PD_VAL & ZQ_HW_PU_VAL will be driven to I/O pads resistor controls. 1 Fields ZQ_SW_PD_VAL & ZQ_SW_PU_VAL will be driven to I/O pads resistor controls.
12 ZQ_SW_PD	ZQ software PU/PD calibration. This bit determines the calibration stage (PU or PD). 0 PU resistor calibration 1 PD resistor calibration
11–7 ZQ_SW_PD_VAL	ZQ software pull-down resistence.This field determines the value of the PD resistor during SW ZQ calibration. 00000 Max. resistance. 11111 Min. resistance.

Table continues on the next page...

MMDCx_MPZQSWCTRL field descriptions (continued)

Field	Description
6–2 ZQ_SW_PU_VAL	ZQ software pull-up resistence.This field determines the value of the PU resistor during SW ZQ calibration. 00000 Min. resistance. 11111 Max. resistance.
1 ZQ_SW_RES	ZQ software calibration result. This bit reflects the ZQ calibration voltage comparator value. 0 Current ZQ calibration voltage is less than VDD/2. 1 Current ZQ calibration voltage is more than VDD/2
0 ZQ_SW_FOR	ZQ SW calibration enable. This bit when asserted enables ZQ SW calibration. HW negates this bit upon completion of the ZQ SW calibration. Upon negation of this bit the ZQ SW calibration result (i.e ZQ_SW_RES) is valid

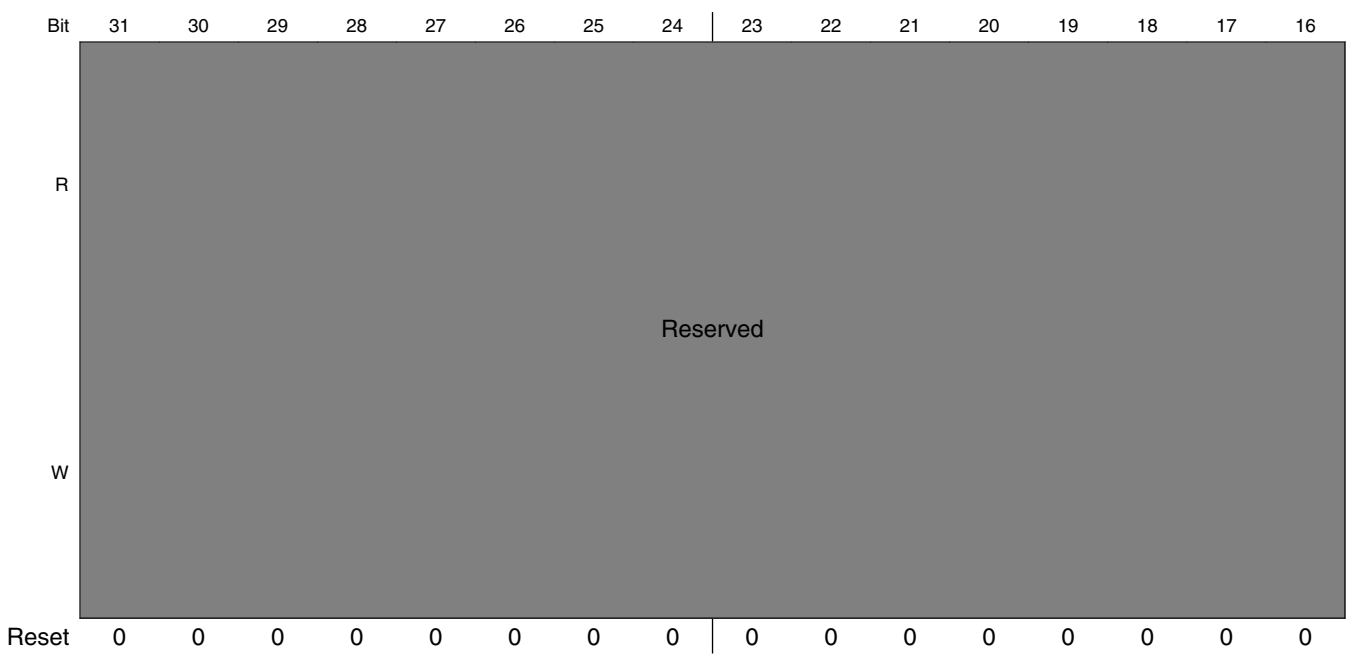
44.12.33 MMDC PHY Write Leveling Configuration and Error Status Register (MMDCx_MPWLGCR)

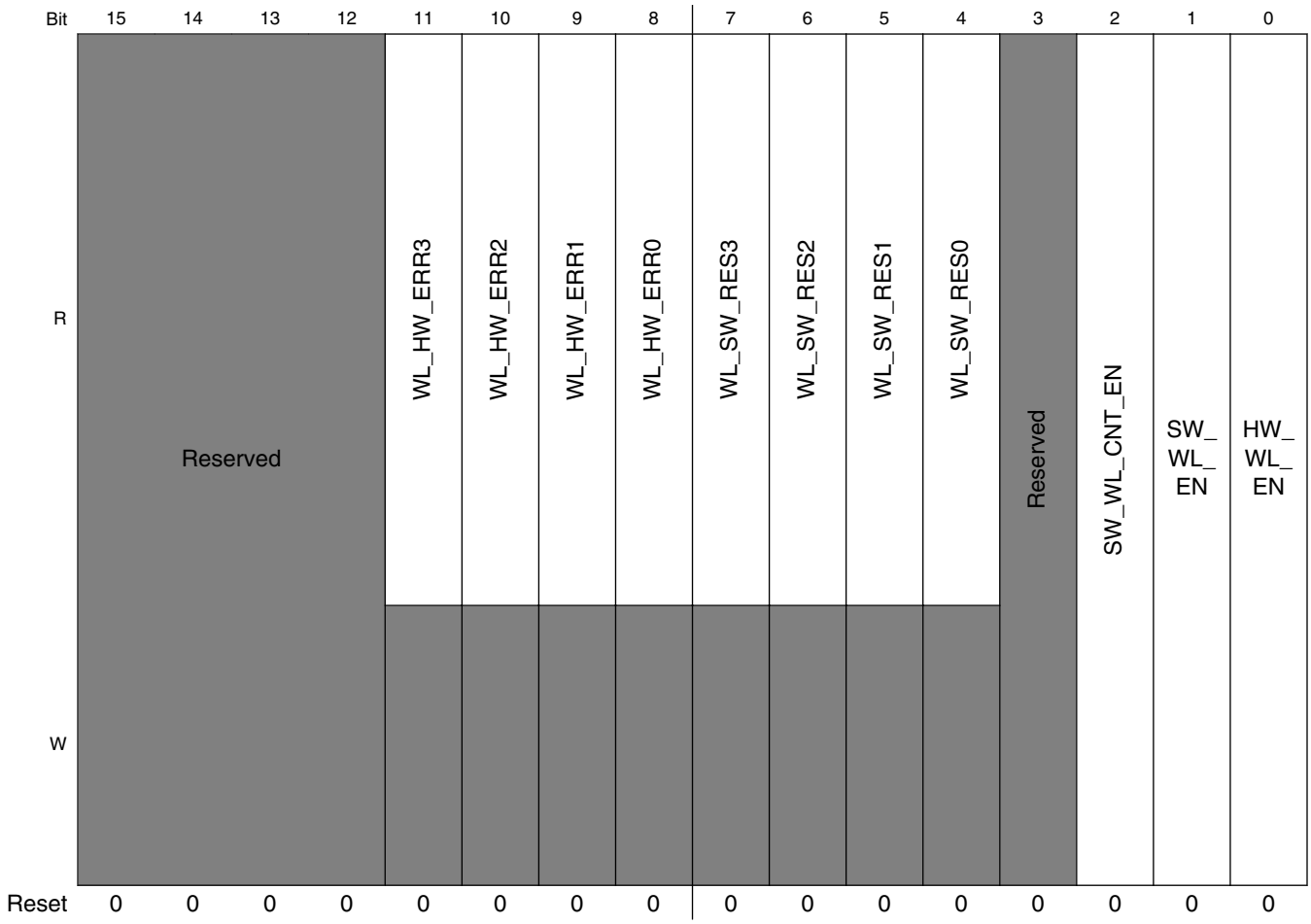
Supported Mode Of Operations:

For Channel 0: All

For Channel 1: DDR3_x64, LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 808h offset





MMDCx_MPWLGCR field descriptions

Field	Description
31–12 -	This field is reserved. Reserved
11 WL_HW_ERR3	Byte3 write-leveling HW calibration error. This bit is asserted when an error was found on byte3 during write-leveling HW calibration. This bit is valid only upon completion of the write-leveling HW calibration (i.e HW_WL_EN bit is de-asserted) 0 No error was found on byte3 during write-leveling HW calibration. 1 An error was found on byte3 during write-leveling HW calibration.
10 WL_HW_ERR2	Byte2 write-leveling HW calibration error. This bit is asserted when an error was found on byte2 during write-leveling HW calibration. This bit is valid only upon completion of the write-leveling HW calibration (i.e HW_WL_EN bit is de-asserted) 0 No error was found on byte2 during write-leveling HW calibration. 1 An error was found on byte2 during write-leveling HW calibration.
9 WL_HW_ERR1	Byte1 write-leveling HW calibration error. This bit is asserted when an error was found on byte1 during write-leveling HW calibration.

Table continues on the next page...

MMDCx_MPWLGCR field descriptions (continued)

Field	Description
	This bit is valid only upon completion of the write-leveling HW calibration (i.e HW_WL_EN bit is de-asserted) 0 No error was found on byte1 during write-leveling HW calibration. 1 An error was found on byte1 during write-leveling HW calibration.
8 WL_HW_ERR0	Byte0 write-leveling HW calibration error. This bit is asserted when an error was found on byte0 during write-leveling HW calibration. This bit is valid only upon completion of the write-leveling HW calibration (i.e HW_WL_EN bit is de-asserted) 0 No error was found on byte0 during write-leveling HW calibration. 1 An error was found on byte0 during write-leveling HW calibration.
7 WL_SW_RES3	Byte3 write-leveling software result. This bit reflects the value that is driven by the DDR device on DQ24 during SW write-leveling. 0 DQS3 sampled low CK during SW write-leveling. 1 DQS3 sampled high CK during SW write-leveling.
6 WL_SW_RES2	Byte2 write-leveling software result. This bit reflects the value that is driven by the DDR device on DQ16 during SW write-leveling. 0 DQS2 sampled low CK during SW write-leveling. 1 DQS2 sampled high CK during SW write-leveling.
5 WL_SW_RES1	Byte1 write-leveling software result. This bit reflects the value that is driven by the DDR device on DQ8 during SW write-leveling. 0 DQS1 sampled low CK during SW write-leveling. 1 DQS1 sampled high CK during SW write-leveling.
4 WL_SW_RES0	Byte0 write-leveling software result. This bit reflects the value that is driven by the DDR device on DQ0 during SW write-leveling. 0 DQS0 sampled low CK during SW write-leveling. 1 DQS0 sampled high CK during SW write-leveling.
3 -	This field is reserved. Reserved
2 SW_WL_CNT_EN	SW write-leveling count down enable.This bit when asserted set a certain delay of (25+15) cycles from the setting of SW_WL_EN and before driving the DQS to the DDR device.This bit should be asserted before the first SW write-leveling request and after issuing the write leveling MRS command 0 MMDC doesn't count 25+15 cycles before issuing write-leveling DQS. 1 MMDC counts 25+15 cycles before issuing write-leveling DQS.
1 SW_WL_EN	Write-Leveling SW enable. If this bit is asserted then the MMDC will perform one write-leveling iteration with the DDR device (assuming that Write-Leveling procedure is already enabled in the DDR device through MRS command). HW negate this bit upon completion of the SW write-leveling. Negation of this bit also points that the write-leveling SW calibration result is valid **NOTE:** If this bit and the SW_WL_CNT_EN are enabled the MMDC counts 25 + 15 cycles before issuing the SW write-leveling DQS.
0 HW_WL_EN	Write-Leveling HW (automatic) enable. If this bit is asserted then the MMDC will perform the whole Write-Leveling sequence with the DDR device (assuming that Write-Leveling procedure is already enabled in the DDR device through MRS command). HW negates this bit upon completion of the HW write-leveling. Negation of this bit also points that the write-leveling HW calibration results are valid

Table continues on the next page...

MMDCx_MPWLGCR field descriptions (continued)

Field	Description
	NOTE: Before issuing the first DQS the MMDC counts 25 + 15 cycles automatically as required by the standard.

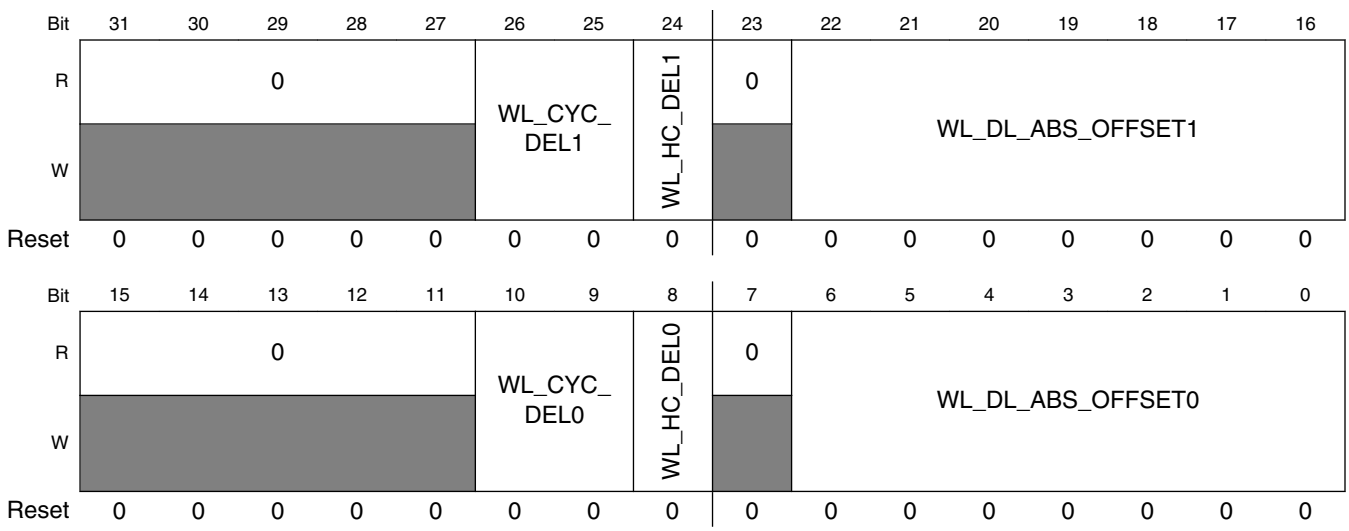
44.12.34 MMDC PHY Write Leveling Delay Control Register 0 (MMDCx_MPWLDECTRL0)

Supported Mode OF Operations:

For Channel 0: All

For Channel 1: DDR3_x64, LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 80Ch offset



MMDCx_MPWLDECTRL0 field descriptions

Field	Description
31–27 Reserved	This read-only field is reserved and always has the value 0.
26–25 WL_CYC_DEL1	Write leveling cycle delay for Byte 1. This field indicates whether a delay of 1 or 2 cycles between CK and write DQS is added to the delay that is indicated in the associated WR_DL_ABS_OFFSET and WL_HC_DEL. So the total delay is the sum of (WL_DL_ABS_OFFSET/256*cycle) + (WL_HC_DEL*half cycle) + (WL_CYC_DEL*cycle). When both SW write-leveling is enabled (i.e SW_WL_EN = 1) or HW write-leveling is enabled (i.e HW_WL_EN = 1) then this value will be taken as is and will be added to the associated delay that is configured in WL_DL_OFFSET and WL_HC_DEL. Note that in HW write-leveling this field is not used for indication, as in WL_DL_OFFSET and WL_HC_DEL, but for configuration. 0 No delay is added.

Table continues on the next page...

MMDCx_MPWLDECTRL0 field descriptions (continued)

Field	Description
	1 1 cycle delay is added. 2 2 cycles delay is added. 3 Reserved.
24 WL_HC_DEL1	Write leveling half cycle delay for Byte 1. This field indicates whether a delay of half cycle between CK and write DQS is added to the delay that is indicated in the associated WR_DL_ABS_OFFSET and WL_CYC_DEL. So the total delay is the sum of (WL_DL_ABS_OFFSET/256*cycle) + (WL_HC_DEL*half cycle) + (WL_CYC_DEL*cycle). When SW write-leveling is enabled (i.e SW_WL_EN = 1) then this value will be taken as is and will be added to the associated delay that is configured in WL_DL_OFFSET and WL_CYC_DEL. When HW write-leveling is enabled (i.e HW_WL_EN = 1) then this value will indicate (status) whether a delay of half cycle was added or not to the associated WL_DL_OFFSET and WL_CYC_DEL. 0 No delay is added. 1 Half cycle delay is added.
23 Reserved	This read-only field is reserved and always has the value 0.
22–16 WL_DL_ABS_ OFFSET1	Absolute write-leveling delay offset for Byte 1. This field indicates the absolute delay between CK and write DQS of Byte1 with fractions of a clock period and up to half cycle. This value is process and frequency independent. The value of the delay can be calculated using the following equation (WR_DL_ABS_OFFSET1 / 256) * clock period When SW write-leveling is enabled (i.e SW_WL_EN = 1) then this value will be taken as is to the associated delay-line. When HW write-leveling is enabled (i.e HW_WL_EN = 1) then this value will indicate (status) the value that is taken to the associated delay-line at the end of the write-leveling calibration. **NOTE:** The delay-line has a resolution that may vary between device to device, therefore is some cases an increment of the delay by 1 step may be smaller than the delay-line resolution.
15–11 Reserved	This read-only field is reserved and always has the value 0.
10–9 WL_CYC_DEL0	Write leveling cycle delay for Byte 0. This field indicates whether a delay of 1 or 2 cycles between CK and write DQS is added to the delay that is indicated in the associated WR_DL_ABS_OFFSET and WL_HC_DEL. So the total delay is the sum of (WL_DL_ABS_OFFSET/256*cycle) + (WL_HC_DEL*half cycle) + (WL_CYC_DEL*cycle). When both SW write-leveling is enabled (i.e SW_WL_EN = 1) or HW write-leveling is enabled (i.e HW_WL_EN = 1) then this value will be taken as is and will be added to the associated delay that is configured in WL_DL_OFFSET and WL_HC_DEL. Note that in HW write-leveling this field is not used for indication, as in WL_DL_OFFSET and WL_HC_DEL, but for configuration. 0 No delay is added. 1 1 cycle delay is added. 2 2 cycles delay is added. 3 Reserved.
8 WL_HC_DEL0	Write leveling half cycle delay for Byte 0. This field indicates whether a delay of half cycle between CK and write DQS is added to the delay that is indicated in the associated WR_DL_ABS_OFFSET and WL_CYC_DEL. So the total delay is the sum of (WL_DL_ABS_OFFSET/256*cycle) + (WL_HC_DEL*half cycle) + (WL_CYC_DEL*cycle).

Table continues on the next page...

MMDCx_MPWLDECTRL0 field descriptions (continued)

Field	Description
	When SW write-leveling is enabled (i.e SW_WL_EN = 1) then this value will be taken as is and will be added to the associated delay that is configured in WL_DL_OFFSET and WL_CYC_DEL. When HW write-leveling is enabled (i.e HW_WL_EN = 1) then this value will indicate (status) whether a delay of half cycle was added or not to the associated WL_DL_OFFSET and WL_CYC_DEL. 0 No delay is added. 1 Half cycle delay is added.
7 Reserved	This read-only field is reserved and always has the value 0.
WL_DL_ABS_OFFSET0	Absolute write-leveling delay offset for Byte 0. This field indicates the absolute delay between CK and write DQS of Byte0 with fractions of a clock period and up to half cycle. This value is process and frequency independent. The value of the delay can be calculated using the following equation (WR_DL_ABS_OFFSET1 / 256) * clock period When SW write-leveling is enabled (i.e SW_WL_EN = 1) then this value will be taken as is to the associated delay-line. When HW write-leveling is enabled (i.e HW_WL_EN = 1) then this value will indicate (status) the value that is taken to the associated delay-line at the end of the write-leveling calibration. **NOTE:** The delay-line has a resolution that may vary between device to device, therefore is some cases an increment of the delay by 1 step may be smaller than the delay-line resolution.

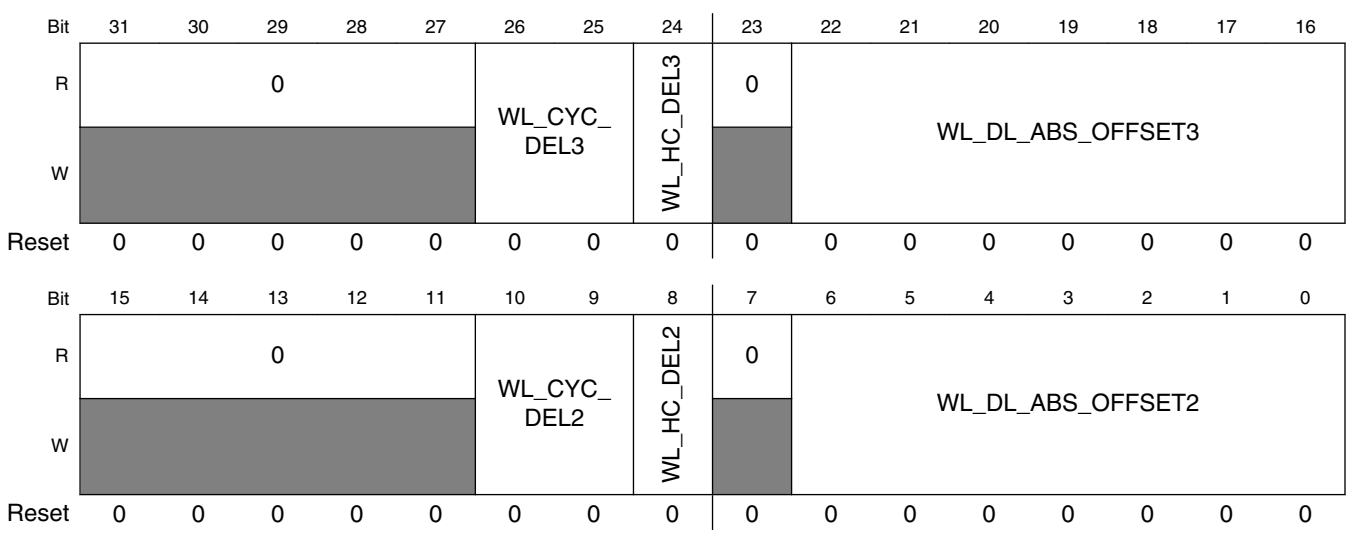
44.12.35 MMDC PHY Write Leveling Delay Control Register 1 (MMDCx_MPWLDECTRL1)

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: DDR3_x64, LP2_2ch_x32

Address: Base address + 810h offset



MMDCx_MPWLDECTRL1 field descriptions

Field	Description
31–27 Reserved	This read-only field is reserved and always has the value 0.
26–25 WL_CYC_DEL3	Write leveling cycle delay for Byte 3. This field indicates whether a delay of 1 or 2 cycles between CK and write DQS is added to the delay that is indicated in the associated WL_DL_ABS_OFFSET and WL_HC_DEL. So the total delay is the sum of (WL_DL_ABS_OFFSET/256*cycle) + (WL_HC_DEL*half cycle) + (WL_CYC_DEL*cycle). When both SW write-leveling is enabled (i.e SW_WL_EN = 1) or HW write-leveling is enabled (i.e HW_WL_EN = 1) then this value will be taken as is and will be added to the associated delay that is configured in WL_DL_OFFSET and WL_HC_DEL. Note that in HW write-leveling this field is not used for indication, as in WL_DL_OFFSET and WL_HC_DEL, but for configuration. 0 No delay is added. 1 1 cycle delay is added. 2 2 cycles delay is added. 3 Reserved.
24 WL_HC_DEL3	Write leveling half cycle delay for Byte 3. This field indicates whether a delay of half cycle between CK and write DQS is added to the delay that is indicated in the associated WL_DL_ABS_OFFSET and WL_CYC_DEL. So the total delay is the sum of (WL_DL_ABS_OFFSET/256*cycle) + (WL_HC_DEL*half cycle) + (WL_CYC_DEL*cycle). When SW write-leveling is enabled (i.e SW_WL_EN = 1) then this value will be taken as is and will be added to the associated delay that is configured in WL_DL_OFFSET and WL_CYC_DEL. When HW write-leveling is enabled (i.e HW_WL_EN = 1) then this value will indicate (status) whether a delay of half cycle was added or not to the associated WL_DL_OFFSET and WL_CYC_DEL. 0 No delay is added. 1 Half cycle delay is added.
23 Reserved	This read-only field is reserved and always has the value 0.
22–16 WL_DL_ABS_ OFFSET3	Absolute write-leveling delay offset for Byte 3. This field indicates the absolute delay between CK and write DQS of Byte3 with fractions of a clock period and up to half cycle. This value is process and frequency independent. The value of the delay can be calculated using the following equation (WL_DL_ABS_OFFSET3 / 256) * clock period When SW write-leveling is enabled (i.e SW_WL_EN = 1) then this value will be taken as is to the associated delay-line. When HW write-leveling is enabled (i.e HW_WL_EN = 1) then this value will indicate (status) the value that is taken to the associated delay-line at the end of the write-leveling calibration. **NOTE:** The delay-line has a resolution that may vary between device to device, therefore is some cases an increment of the delay by 1 step may be smaller than the delay-line resolution.
15–11 Reserved	This read-only field is reserved and always has the value 0.
10–9 WL_CYC_DEL2	Write leveling cycle delay for Byte 2. This field indicates whether a delay of 1 or 2 cycles between CK and write DQS is added to the delay that is indicated in the associated WR_DL_ABS_OFFSET and WL_HC_DEL. So the total delay is the sum of (WL_DL_ABS_OFFSET/256*cycle) + (WL_HC_DEL*half cycle) + (WL_CYC_DEL*cycle). When both SW write-leveling is enabled (i.e SW_WL_EN = 1) or HW write-leveling is enabled (i.e HW_WL_EN = 1) then this value will be taken as is and will be added to the associated delay that is configured in WL_DL_OFFSET and WL_HC_DEL.

Table continues on the next page...

MMDCx_MPWLDECTRL1 field descriptions (continued)

Field	Description
	Note that in HW write-leveling this field is not used for indication, as in WL_DL_OFFSET and WL_HC_DEL, but for configuration. 0 No delay is added. 1 1 cycle delay is added. 2 2 cycles delay is added. 3 Reserved.
8 WL_HC_DEL2	Write leveling half cycle delay for Byte 2. This field indicates whether a delay of half cycle between CK and write DQS is added to the delay that is indicated in the associated WR_DL_ABS_OFFSET and WL_CYC_DEL. So the total delay is the sum of (WL_DL_ABS_OFFSET/256*cycle) + (WL_HC_DEL*half cycle) + (WL_CYC_DEL*cycle). When SW write-leveling is enabled (i.e SW_WL_EN = 1) then this value will be taken as is and will be added to the associated delay that is configured in WL_DL_OFFSET and WL_CYC_DEL. When HW write-leveling is enabled (i.e HW_WL_EN = 1) then this value will indicate (status) whether a delay of half cycle was added or not to the associated WL_DL_OFFSET and WL_CYC_DEL. 0 No delay is added. 1 Half cycle delay is added.
7 Reserved	This read-only field is reserved and always has the value 0.
WL_DL_ABS_ OFFSET2	Absolute write-leveling delay offset for Byte 2. This field indicates the absolute delay between CK and write DQS of Byte1 with fractions of a clock period and up to half cycle. This value is process and frequency independent. The value of the delay can be calculated using the following equation (WR_DL_ABS_OFFSET2 / 256) * clock period When SW write-leveling is enabled (i.e SW_WL_EN = 1) then this value will be taken as is to the associated delay-line. When HW write-leveling is enabled (i.e HW_WL_EN = 1) then this value will indicate (status) the value that is taken to the associated delay-line at the end of the write-leveling calibration. **NOTE:** The delay-line has a resolution that may vary between device to device, therefore is some cases an increment of the delay by 1 step may be smaller than the delay-line resolution.

44.12.36 MMDC PHY Write Leveling delay-line Status Register (MMDCx_MPWLDLST)

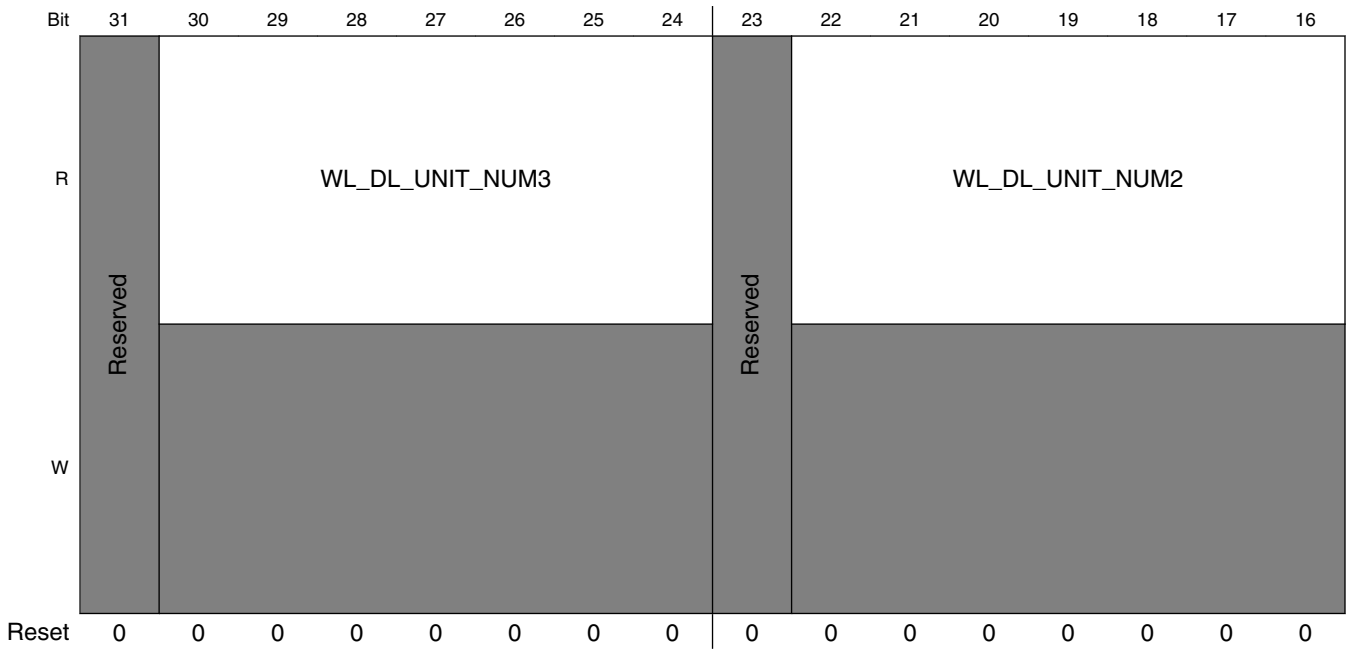
This register holds the status of the four write leveling delay-lines.

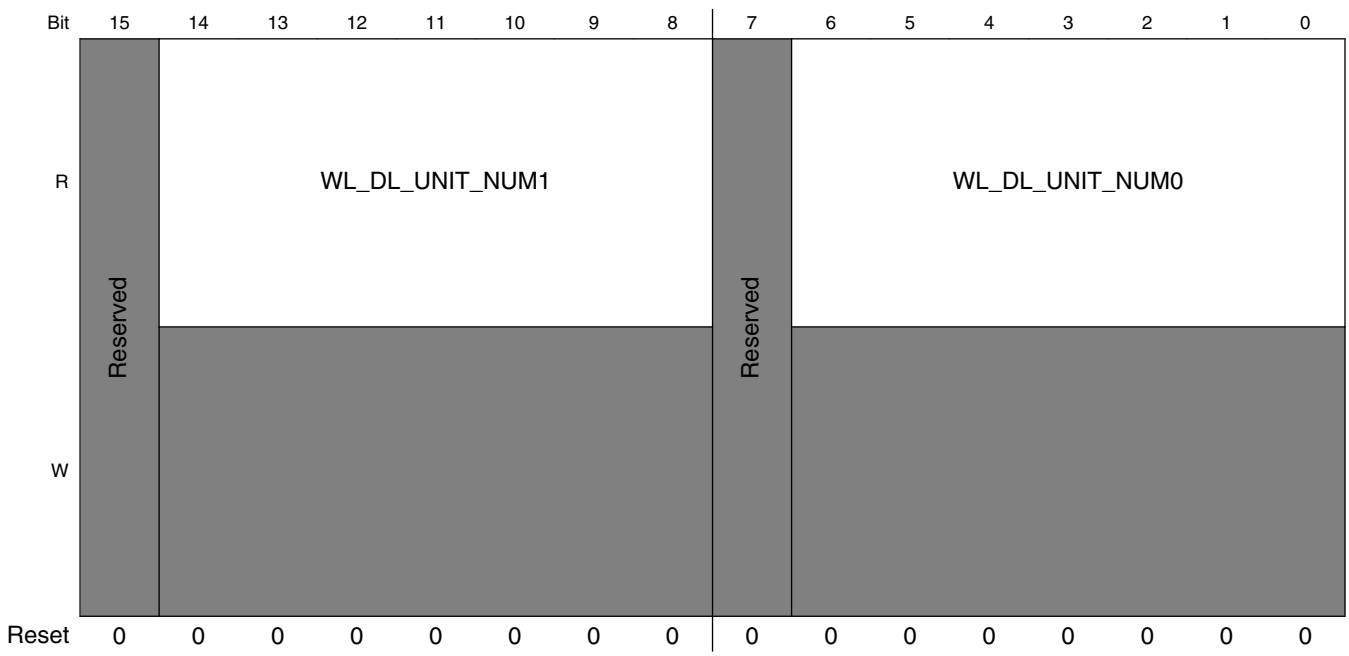
Supported Mode Of Operations:

For Channel 0: All

For Channel 1: DDR3_x64, LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 814h offset





MMDCx_MPWLDLST field descriptions

Field	Description
31 -	This field is reserved. Reserved
30–24 WL_DL_UNIT_NUM3	This field reflects the number of delay units that are actually used by write leveling delay-line 3.
23 -	This field is reserved. Reserved

Table continues on the next page...

MMDCx_MPWLDLST field descriptions (continued)

Field	Description
22–16 WL_DL_UNIT_NUM2	This field reflects the number of delay units that are actually used by write leveling delay-line 2.
15 -	This field is reserved. Reserved
14–8 WL_DL_UNIT_NUM1	This field reflects the number of delay units that are actually used by write leveling delay-line 1.
7 -	This field is reserved. Reserved
WL_DL_UNIT_NUM0	This field reflects the number of delay units that are actually used by write leveling delay-line 0.

44.12.37 MMDC PHY ODT control register (MMDCx_MPODTCTRL)

NOTE

In LPDDR2 mode this register should be cleared, so no termination will be activated

Supported Mode Of Operations:

For Channel 0: DDR3_x16, DDR3_x32, DDR3_x64

For Channel 1: DDR3_x64

Address: Base address + 818h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0													ODT3_INT_RES		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	ODT2_INT_RES			0	ODT1_INT_RES			0	ODT0_INT_RES			ODT_RD_ACT_EN	ODT_RD_PAS_EN	ODT_WR_ACT_EN	ODT_WR_PAS_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDCx_MPODTCTRL field descriptions

Field	Description
31–19 Reserved	This read-only field is reserved and always has the value 0.
18–16 ODT3_INT_RES	On chip ODT byte3 resistor - This field determines the Rtt_Nom of the on chip ODT byte3 resistor during read accesses. 000 Rtt_Nom Disabled. 001 Rtt_Nom 120 Ohm 010 Rtt_Nom 60 Ohm 011 Rtt_Nom 40 Ohm 100 Rtt_Nom 30 Ohm 101 Rtt_Nom 24 Ohm 110 Rtt_Nom 20 Ohm 111 Rtt_Nom 17 Ohm
15 Reserved	This read-only field is reserved and always has the value 0.
14–12 ODT2_INT_RES	On chip ODT byte2 resistor - This field determines the Rtt_Nom of the on chip ODT byte2 resistor during read accesses. 000 Rtt_Nom Disabled. 001 Rtt_Nom 120 Ohm 010 Rtt_Nom 60 Ohm 011 Rtt_Nom 40 Ohm 100 Rtt_Nom 30 Ohm 101 Rtt_Nom 24 Ohm 110 Rtt_Nom 20 Ohm 111 Rtt_Nom 17 Ohm
11 Reserved	This read-only field is reserved and always has the value 0.
10–8 ODT1_INT_RES	On chip ODT byte1 resistor - This field determines the Rtt_Nom of the on chip ODT byte1 resistor during read accesses. 0000 Rtt_Nom Disabled. 001 Rtt_Nom 120 Ohm 010 Rtt_Nom 60 Ohm 011 Rtt_Nom 40 Ohm 100 Rtt_Nom 30 Ohm 101 Rtt_Nom 24 Ohm 110 Rtt_Nom 20 Ohm 111 Rtt_Nom 17 Ohm
7 Reserved	This read-only field is reserved and always has the value 0.
6–4 ODT0_INT_RES	On chip ODT byte0 resistor - This field determines the Rtt_Nom of the on chip ODT byte0 resistor during read accesses. 000 Rtt_Nom Disabled. 001 Rtt_Nom 120 Ohm 010 Rtt_Nom 60 Ohm 011 Rtt_Nom 40 Ohm

Table continues on the next page...

MMDCx_MPODTCTRL field descriptions (continued)

Field	Description
	100 Rtt_Nom 30 Ohm 101 Rtt_Nom 24 Ohm 110 Rtt_Nom 20 Ohm 111 Rtt_Nom 17 Ohm
3 ODT_RD_ACT_EN	Active read CS ODT enable. The bit determines if ODT pin of the active CS will be asserted during read accesses. 0 Active CS ODT pin is disabled during read access. 1 Active CS ODT pin is enabled during read access.
2 ODT_RD_PAS_EN	Inactive read CS ODT enable. The bit determines if ODT pin of the inactive CS will be asserted during read accesses. 0 Inactive CS ODT pin is disabled during read accesses to other CS. 1 Inactive CS ODT pin is enabled during read accesses to other CS.
1 ODT_WR_ACT_EN	Active write CS ODT enable. The bit determines if ODT pin of the active CS will be asserted during write accesses. 0 Active CS ODT pin is disabled during write access. 1 Active CS ODT pin is enabled during write access.
0 ODT_WR_PAS_EN	Inactive write CS ODT enable. The bit determines if ODT pin of the inactive CS will be asserted during write accesses. 0 Inactive CS ODT pin is disabled during write accesses to other CS. 1 Inactive CS ODT pin is enabled during write accesses to other CS.

44.12.38 MMDC PHY Read DQ Byte0 Delay Register (MMDCx_MPRDDQBY0DL)

This register is used to add fine-tuning adjustment to every bit in the read DQ byte0 relative to the read DQS. This delay is in addition to the read data calibration. If operating in 64-bit mode, there is an identical register that is mapped at the second base address.

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: DDR3_x64, LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 81Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	rd_dq7_del			0	rd_dq6_del			0	rd_dq5_del			0	rd_dq4_del		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	rd_dq3_del			0	rd_dq2_del			0	rd_dq1_del			0	rd_dq0_del		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDCx_MPRDDQBY0DL field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30–28 rd_dq7_del	Read dqs0 to dq7 delay fine-tuning. This field holds the number of delay units that are added to dq7 relative to dqs0. 000 No change in dq7 delay 001 Add dq7 delay of 1 delay unit 010 Add dq7 delay of 2 delay units. 011 Add dq7 delay of 3 delay units. 100 Add dq7 delay of 4 delay units. 101 Add dq7 delay of 5 delay units. 110 Add dq7 delay of 6 delay units. 111 Add dq7 delay of 7 delay units.
27 Reserved	This read-only field is reserved and always has the value 0.
26–24 rd_dq6_del	Read dqs0 to dq6 delay fine-tuning. This field holds the number of delay units that are added to dq6 relative to dqs0. 000 No change in dq6 delay 001 Add dq6 delay of 1 delay unit 010 Add dq6 delay of 2 delay units. 011 Add dq6 delay of 3 delay units. 100 Add dq6 delay of 4 delay units. 101 Add dq6 delay of 5 delay units. 110 Add dq6 delay of 6 delay units. 111 Add dq6 delay of 7 delay units.
23 Reserved	This read-only field is reserved and always has the value 0.
22–20 rd_dq5_del	Read dqs0 to dq5 delay fine-tuning. This field holds the number of delay units that are added to dq5 relative to dqs0. 000 No change in dq5 delay 001 Add dq5 delay of 1 delay unit 010 Add dq5 delay of 2 delay units. 011 Add dq5 delay of 3 delay units. 100 Add dq5 delay of 4 delay units. 101 Add dq5 delay of 5 delay units. 110 Add dq5 delay of 6 delay units. 111 Add dq5 delay of 7 delay units.
19 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

MMDCx_MPRDDQBY0DL field descriptions (continued)

Field	Description
18–16 rd_dq4_del	Read dqs0 to dq4 delay fine-tuning. This field holds the number of delay units that are added to dq4 relative to dqs0. 000 No change in dq4 delay 001 Add dq4 delay of 1 delay unit 010 Add dq4 delay of 2 delay units. 011 Add dq4 delay of 3 delay units. 100 Add dq4 delay of 4 delay units. 101 Add dq4 delay of 5 delay units. 110 Add dq4 delay of 6 delay units. 111 Add dq4 delay of 7 delay units.
15 Reserved	This read-only field is reserved and always has the value 0.
14–12 rd_dq3_del	Read dqs0 to dq3 delay fine-tuning. This field holds the number of delay units that are added to dq3 relative to dqs0. 000 No change in dq3 delay 001 Add dq3 delay of 1 delay unit 010 Add dq3 delay of 2 delay units. 011 Add dq3 delay of 3 delay units. 100 Add dq3 delay of 4 delay units. 101 Add dq3 delay of 5 delay units. 110 Add dq3 delay of 6 delay units. 111 Add dq3 delay of 7 delay units.
11 Reserved	This read-only field is reserved and always has the value 0.
10–8 rd_dq2_del	Read dqs0 to dq2 delay fine-tuning. This field holds the number of delay units that are added to dq2 relative to dqs0. 000 No change in dq2 delay 001 Add dq2 delay of 1 delay unit 010 Add dq2 delay of 2 delay units. 011 Add dq2 delay of 3 delay units. 100 Add dq2 delay of 4 delay units. 101 Add dq2 delay of 5 delay units. 110 Add dq2 delay of 6 delay units. 111 Add dq2 delay of 7 delay units.
7 Reserved	This read-only field is reserved and always has the value 0.
6–4 rd_dq1_del	Read dqs0 to dq1 delay fine-tuning. This field holds the number of delay units that are added to dq1 relative to dqs0. 000 No change in dq1 delay 001 Add dq1 delay of 1 delay unit 010 Add dq1 delay of 2 delay units. 011 Add dq1 delay of 3 delay units. 100 Add dq1 delay of 4 delay units. 101 Add dq1 delay of 5 delay units.

Table continues on the next page...

MMDCx_MPRDDQBY0DL field descriptions (continued)

Field	Description
	110 Add dq1 delay of 6 delay units. 111 Add dq1 delay of 7 delay units.
3 Reserved	This read-only field is reserved and always has the value 0.
rd_dq0_del	Read dqs0 to dq0 delay fine-tuning. This field holds the number of delay units that are added to dq0 relative to dqs0. 000 No change in dq0 delay 001 Add dq0 delay of 1 delay unit 010 Add dq0 delay of 2 delay units. 011 Add dq0 delay of 3 delay units. 100 Add dq0 delay of 4 delay units. 101 Add dq0 delay of 5 delay units. 110 Add dq0 delay of 6 delay units. 111 Add dq0 delay of 7 delay units.

44.12.39 MMDC PHY Read DQ Byte1 Delay Register (MMDCx_MPRDDQBY1DL)

This register is used to add fine-tuning adjustment to every bit in the read DQ byte1 relative to the read DQS

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: DDR3_x64, LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 820h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	rd_dq15_del			0	rd_dq14_del			0	rd_dq13_del			0	rd_dq12_del		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	rd_dq11_del			0	rd_dq10_del			0	rd_dq9_del			0	rd_dq8_del		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDCx_MPRDDQBY1DL field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

MMDCx_MPRDDQBY1DL field descriptions (continued)

Field	Description
30–28 rd_dq15_del	Read dqs1 to dq15 delay fine-tuning. This field holds the number of delay units that are added to dq15 relative to dqs1. 000 No change in dq15 delay 001 Add dq15 delay of 1 delay unit 010 Add dq15 delay of 2 delay units. 011 Add dq15 delay of 3 delay units. 100 Add dq15 delay of 4 delay units. 101 Add dq15 delay of 5 delay units. 110 Add dq15 delay of 6 delay units. 111 Add dq15 delay of 7 delay units.
27 Reserved	This read-only field is reserved and always has the value 0.
26–24 rd_dq14_del	Read dqs1 to dq14 delay fine-tuning. This field holds the number of delay units that are added to dq14 relative to dqs1. 000 No change in dq14 delay 001 Add dq14 delay of 1 delay unit 010 Add dq14 delay of 2 delay units. 011 Add dq14 delay of 3 delay units. 100 Add dq14 delay of 4 delay units. 101 Add dq14 delay of 5 delay units. 110 Add dq14 delay of 6 delay units. 111 Add dq14 delay of 7 delay units.
23 Reserved	This read-only field is reserved and always has the value 0.
22–20 rd_dq13_del	Read dqs1 to dq13 delay fine-tuning. This field holds the number of delay units that are added to dq13 relative to dqs1. 000 No change in dq13 delay 001 Add dq13 delay of 1 delay unit 010 Add dq13 delay of 2 delay units. 011 Add dq13 delay of 3 delay units. 100 Add dq13 delay of 4 delay units. 101 Add dq13 delay of 5 delay units. 110 Add dq13 delay of 6 delay units. 111 Add dq13 delay of 7 delay units.
19 Reserved	This read-only field is reserved and always has the value 0.
18–16 rd_dq12_del	Read dqs1 to dq12 delay fine-tuning. This field holds the number of delay units that are added to dq12 relative to dqs1. 000 No change in dq12 delay 001 Add dq12 delay of 1 delay unit 010 Add dq12 delay of 2 delay units. 011 Add dq12 delay of 3 delay units. 100 Add dq12 delay of 4 delay units. 101 Add dq12 delay of 5 delay units.

Table continues on the next page...

MMDCx_MPRDDQBY1DL field descriptions (continued)

Field	Description
	110 Add dq12 delay of 6 delay units. 111 Add dq12 delay of 7 delay units.
15 Reserved	This read-only field is reserved and always has the value 0.
14–12 rd_dq11_del	Read dqs1 to dq11 delay fine-tuning. This field holds the number of delay units that are added to dq11 relative to dqs1. 000 No change in dq11 delay 001 Add dq11 delay of 1 delay unit 010 Add dq11 delay of 2 delay units. 011 Add dq11 delay of 3 delay units. 100 Add dq11 delay of 4 delay units. 101 Add dq11 delay of 5 delay units. 110 Add dq11 delay of 6 delay units. 111 Add dq11 delay of 7 delay units.
11 Reserved	This read-only field is reserved and always has the value 0.
10–8 rd_dq10_del	Read dqs1 to dq10 delay fine-tuning. This field holds the number of delay units that are added to dq10 relative to dqs1. 000 No change in dq10 delay 001 Add dq10 delay of 1 delay unit 010 Add dq10 delay of 2 delay units. 011 Add dq10 delay of 3 delay units. 100 Add dq10 delay of 4 delay units. 101 Add dq10 delay of 5 delay unit 110 Add dq10 delay of 6 delay units. 111 Add dq10 delay of 7 delay units.
7 Reserved	This read-only field is reserved and always has the value 0.
6–4 rd_dq9_del	Read dqs1 to dq9 delay fine-tuning. This field holds the number of delay units that are added to dq9 relative to dqs1. 000 No change in dq9 delay 001 Add dq9 delay of 1 delay unit 010 Add dq9 delay of 2 delay units. 011 Add dq9 delay of 3 delay units. 100 Add dq9 delay of 4 delay units. 101 Add dq9 delay of 5 delay units. 110 Add dq9 delay of 6 delay units. 111 Add dq9 delay of 7 delay units.
3 Reserved	This read-only field is reserved and always has the value 0.
rd_dq8_del	Read dqs1 to dq8 delay fine-tuning. This field holds the number of delay units that are added to dq8 relative to dqs1. 000 No change in dq8 delay 001 Add dq8 delay of 1 delay unit

Table continues on the next page...

MMDCx_MPRDDQBY1DL field descriptions (continued)

Field	Description
	010 Add dq8 delay of 2 delay units. 011 Add dq8 delay of 3 delay units. 100 Add dq8 delay of 4 delay units. 101 Add dq8 delay of 5 delay units. 110 Add dq8 delay of 6 delay units. 111 Add dq8 delay of 7 delay units.

44.12.40 MMDC PHY Read DQ Byte2 Delay Register (MMDCx_MPRDDQBY2DL)

This register is used to add fine-tuning adjustment to every bit in the read DQ byte2 relative to the read DQS

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: DDR3_x64, LP2_2ch_x32

Address: Base address + 824h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	rd_dq23_del			0	rd_dq22_del			0	rd_dq21_del			0	rd_dq20_del		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	rd_dq19_del			0	rd_dq18_del			0	rd_dq17_del			0	rd_dq16_del		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDCx_MPRDDQBY2DL field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30–28 rd_dq23_del	Read dqs2 to dq23 delay fine-tuning. This field holds the number of delay units that are added to dq23 relative to dqs2. 000 No change in dq23 delay 001 Add dq23 delay of 1 delay unit 010 Add dq23 delay of 2 delay units. 011 Add dq23 delay of 3 delay units. 100 Add dq23 delay of 4 delay units. 101 Add dq23 delay of 5 delay units.

Table continues on the next page...

MMDCx_MPRDDQBY2DL field descriptions (continued)

Field	Description
	110 Add dq23 delay of 6 delay units. 111 Add dq23 delay of 7 delay units.
27 Reserved	This read-only field is reserved and always has the value 0.
26–24 rd_dq22_del	Read dqs2 to dq22 delay fine-tuning. This field holds the number of delay units that are added to dq22 relative to dqs2. 000 No change in dq22 delay 001 Add dq22 delay of 1 delay unit 010 Add dq22 delay of 2 delay units. 011 Add dq22 delay of 3 delay units. 100 Add dq22 delay of 4 delay units. 101 Add dq22 delay of 5 delay units. 110 Add dq22 delay of 6 delay units. 111 Add dq22 delay of 7 delay units.
23 Reserved	This read-only field is reserved and always has the value 0.
22–20 rd_dq21_del	Read dqs2 to dq21 delay fine-tuning. This field holds the number of delay units that are added to dq21 relative to dqs2. 000 No change in dq21 delay 001 Add dq21 delay of 1 delay unit 010 Add dq21 delay of 2 delay units. 011 Add dq21 delay of 3 delay units. 100 Add dq21 delay of 4 delay units. 101 Add dq21 delay of 5 delay units. 110 Add dq21 delay of 6 delay units. 111 Add dq21 delay of 7 delay units.
19 Reserved	This read-only field is reserved and always has the value 0.
18–16 rd_dq20_del	Read dqs2 to dq20 delay fine-tuning. This field holds the number of delay units that are added to dq20 relative to dqs2. 000 No change in dq20 delay 001 Add dq20 delay of 1 delay unit 010 Add dq20 delay of 2 delay units. 011 Add dq20 delay of 3 delay units. 100 Add dq20 delay of 4 delay units. 101 Add dq20 delay of 5 delay units. 110 Add dq20 delay of 6 delay units. 111 Add dq20 delay of 7 delay units.
15 Reserved	This read-only field is reserved and always has the value 0.
14–12 rd_dq19_del	Read dqs2 to dq19 delay fine-tuning. This field holds the number of delay units that are added to dq19 relative to dqs2. 000 No change in dq19 delay 001 Add dq19 delay of 1 delay unit

Table continues on the next page...

MMDCx_MPRDDQBY2DL field descriptions (continued)

Field	Description
	010 Add dq19 delay of 2 delay units. 011 Add dq19 delay of 3 delay units. 100 Add dq19 delay of 4 delay units. 101 Add dq19 delay of 5 delay units. 110 Add dq19 delay of 6 delay units. 111 Add dq19 delay of 7 delay units.
11 Reserved	This read-only field is reserved and always has the value 0.
10–8 rd_dq18_del	Read dqs2 to dq18 delay fine-tuning. This field holds the number of delay units that are added to dq18 relative to dqs2. 000 No change in dq18 delay 001 Add dq18 delay of 1 delay unit 010 Add dq18 delay of 2 delay units. 011 Add dq18 delay of 3 delay units. 100 Add dq18 delay of 4 delay units. 101 Add dq18 delay of 5 delay units. 110 Add dq18 delay of 6 delay units. 111 Add dq18 delay of 7 delay units.
7 Reserved	This read-only field is reserved and always has the value 0.
6–4 rd_dq17_del	Read dqs2 to dq17 delay fine-tuning. This field holds the number of delay units that are added to dq17 relative to dqs2. 000 No change in dq17 delay 001 Add dq17 delay of 1 delay unit 010 Add dq17 delay of 2 delay units. 011 Add dq17 delay of 3 delay units. 100 Add dq17 delay of 4 delay units. 101 Add dq17 delay of 5 delay units. 110 Add dq17 delay of 6 delay units. 111 Add dq17 delay of 7 delay units.
3 Reserved	This read-only field is reserved and always has the value 0.
rd_dq16_del	Read dqs2 to dq16 delay fine-tuning. This field holds the number of delay units that are added to dq16 relative to dqs2. 000 No change in dq16 delay 001 Add dq16 delay of 1 delay unit 010 Add dq16 delay of 2 delay units. 011 Add dq16 delay of 3 delay units. 100 Add dq16 delay of 4 delay units. 101 Add dq16 delay of 5 delay units. 110 Add dq16 delay of 6 delay units. 111 Add dq16 delay of 7 delay units.

44.12.41 MMDC PHY Read DQ Byte3 Delay Register (MMDCx_MPRDDQBY3DL)

This register is used to add fine-tuning adjustment to every bit in the read DQ byte3 relative to the read DQS.

The bit assignments and the bit field descriptions for the register are shown below.

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: DDR3_x64, LP2_2ch_x32

Address: Base address + 828h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	rd_dq31_del			0	rd_dq30_del			0	rd_dq29_del			0	rd_dq28_del		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	rd_dq27_del			0	rd_dq26_del			0	rd_dq25_del			0	rd_dq24_del		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDCx_MPRDDQBY3DL field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30–28 rd_dq31_del	Read dqs3 to dq31 delay fine-tuning. This field holds the number of delay units that are added to dq31 relative to dqs3. 000 No change in dq31 delay 001 Add dq31 delay of 1 delay unit 010 Add dq31 delay of 2 delay units. 011 Add dq31 delay of 3 delay units. 100 Add dq31 delay of 4 delay units. 101 Add dq31 delay of 5 delay units. 110 Add dq31 delay of 6 delay units. 111 Add dq31 delay of 7 delay units.
27 Reserved	This read-only field is reserved and always has the value 0.
26–24 rd_dq30_del	Read dqs3 to dq30 delay fine-tuning. This field holds the number of delay units that are added to dq30 relative to dqs3. 000 No change in dq30 delay 001 Add dq30 delay of 1 delay unit 010 Add dq30 delay of 2 delay units.

Table continues on the next page...

MMDCx_MPRDDQBY3DL field descriptions (continued)

Field	Description
	011 Add dq30 delay of 3 delay units. 100 Add dq30 delay of 4 delay units. 101 Add dq30 delay of 5 delay units. 110 Add dq30 delay of 6 delay units. 111 Add dq30 delay of 7 delay units.
23 Reserved	This read-only field is reserved and always has the value 0.
22–20 rd_dq29_del	Read dqs3 to dq29 delay fine-tuning. This field holds the number of delay units that are added to dq29 relative to dqs3. 000 No change in dq29 delay 001 Add dq29 delay of 1 delay unit 010 Add dq29 delay of 2 delay units. 011 Add dq29 delay of 3 delay units. 100 Add dq29 delay of 4 delay units. 101 Add dq29 delay of 5 delay units. 110 Add dq29 delay of 6 delay units. 111 Add dq29 delay of 7 delay units.
19 Reserved	This read-only field is reserved and always has the value 0.
18–16 rd_dq28_del	Read dqs3 to dq28 delay fine-tuning. This field holds the number of delay units that are added to dq28 relative to dqs3. 000 No change in dq28 delay 001 Add dq28 delay of 1 delay unit 010 Add dq28 delay of 2 delay units. 011 Add dq28 delay of 3 delay units. 100 Add dq28 delay of 4 delay units. 101 Add dq28 delay of 5 delay units. 110 Add dq28 delay of 6 delay units. 111 Add dq28 delay of 7 delay units.
15 Reserved	This read-only field is reserved and always has the value 0.
14–12 rd_dq27_del	Read dqs3 to dq27 delay fine-tuning. This field holds the number of delay units that are added to dq27 relative to dqs3. 000 No change in dq27 delay 001 Add dq27 delay of 1 delay unit 010 Add dq27 delay of 2 delay units. 011 Add dq27 delay of 3 delay units. 100 Add dq27 delay of 4 delay units. 101 Add dq27 delay of 5 delay units. 110 Add dq27 delay of 6 delay units. 111 Add dq27 delay of 7 delay units.
11 Reserved	This read-only field is reserved and always has the value 0.
10–8 rd_dq26_del	Read dqs3 to dq26 delay fine-tuning. This field holds the number of delay units that are added to dq26 relative to dqs3.

Table continues on the next page...

MMDCx_MPRDDQBY3DL field descriptions (continued)

Field	Description
	000 No change in dq26 delay 001 Add dq26 delay of 1 delay unit 010 Add dq26 delay of 2 delay units. 011 Add dq26 delay of 3 delay units. 100 Add dq26 delay of 4 delay units. 101 Add dq26 delay of 5 delay units. 110 Add dq26 delay of 6 delay units. 111 Add dq26 delay of 7 delay units.
7 Reserved	This read-only field is reserved and always has the value 0.
6–4 rd_dq25_del	Read dqs3 to dq25 delay fine-tuning. This field holds the number of delay units that are added to dq25 relative to dqs3. 000 No change in dq25 delay 001 Add dq25 delay of 1 delay unit 010 Add dq25 delay of 2 delay units. 011 Add dq25 delay of 3 delay units. 100 Add dq25 delay of 4 delay units. 101 Add dq25 delay of 5 delay units. 110 Add dq25 delay of 6 delay units. 111 Add dq25 delay of 7 delay units.
3 Reserved	This read-only field is reserved and always has the value 0.
rd_dq24_del	Read dqs3 to dq24 delay fine-tuning. This field holds the number of delay units that are added to dq24 relative to dqs3. 000 No change in dq24 delay 001 Add dq24 delay of 1 delay unit 010 Add dq24 delay of 2 delay units. 011 Add dq24 delay of 3 delay units. 100 Add dq24 delay of 4 delay units. 101 Add dq24 delay of 5 delay units. 110 Add dq24 delay of 6 delay units. 111 Add dq24 delay of 7 delay units.

44.12.42 MMDC PHY Write DQ Byte0 Delay Register (MMDCx_MPWRDQBY0DL)

This register is used to add fine-tuning adjustment to every bit in the write DQ byte0 relative to the write DQS

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: DDR3_x64, LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 82Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	wr_dm0_del		wr_dq7_del		0		wr_dq6_del		0		wr_dq5_del		0		wr_dq4_del	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0		wr_dq3_del		0		wr_dq2_del		0		wr_dq1_del		0		wr_dq0_del	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDCx_MPWRDQBY0DL field descriptions

Field	Description
31–30 wr_dm0_del	Write dm0 delay fine-tuning. This field holds the number of delay units that are added to dm0 relative to dqs0. 00 No change in dm0 delay 01 Add dm0 delay of 1 delay unit. 10 Add dm0 delay of 2 delay units. 11 Add dm0 delay of 3 delay units.
29–28 wr_dq7_del	Write dq7 delay fine-tuning. This field holds the number of delay units that are added to dq7 relative to dqs0. 00 No change in dq7 delay 01 Add dq7 delay of 1 delay unit. 10 Add dq7 delay of 2 delay units. 11 Add dq7 delay of 3 delay units.
27–26 Reserved	This read-only field is reserved and always has the value 0.
25–24 wr_dq6_del	Write dq6 delay fine-tuning. This field holds the number of delay units that are added to dq6 relative to dqs0. 00 No change in dq6 delay 01 Add dq6 delay of 1 delay unit. 10 Add dq6 delay of 2 delay units. 11 Add dq6 delay of 3 delay units.
23–22 Reserved	This read-only field is reserved and always has the value 0.
21–20 wr_dq5_del	Write dq5 delay fine-tuning. This field holds the number of delay units that are added to dq5 relative to dqs0. 00 No change in dq5 delay 01 Add dq5 delay of 1 delay unit. 10 Add dq5 delay of 2 delay units. 11 Add dq5 delay of 3 delay units.
19–18 Reserved	This read-only field is reserved and always has the value 0.
17–16 wr_dq4_del	Write dq4 delay fine-tuning. This field holds the number of delay units that are added to dq4 relative to dqs0. 00 No change in dq4 delay

Table continues on the next page...

MMDCx_MPWRDQBY0DL field descriptions (continued)

Field	Description
	01 Add dq4 delay of 1 delay unit.. 10 Add dq4 delay of 2 delay units. 11 Add dq4 delay of 3 delay units.
15–14 Reserved	This read-only field is reserved and always has the value 0.
13–12 wr_dq3_del	Write dq3 delay fine-tuning. This field holds the number of delay units that are added to dq3 relative to dqs0. 00 No change in dq3 delay 01 Add dq3 delay of 1 delay unit. 10 Add dq3 delay of 2 delay units. 11 Add dq3 delay of 3 delay units.
11–10 Reserved	This read-only field is reserved and always has the value 0.
9–8 wr_dq2_del	Write dq2 delay fine-tuning. This field holds the number of delay units that are added to dq2 relative to dqs0. 00 No change in dq2 delay 01 Add dq2 delay of 1 delay unit. 10 Add dq2 delay of 2 delay units. 11 Add dq2 delay of 3 delay units.
7–6 Reserved	This read-only field is reserved and always has the value 0.
5–4 wr_dq1_del	Write dq1 delay fine-tuning. This field holds the number of delay units that are added to dq1 relative to dqs0. 00 No change in dq1 delay 01 Add dq1 delay of 1 delay unit. 10 Add dq1 delay of 2 delay units. 11 Add dq1 delay of 3 delay units.
3–2 Reserved	This read-only field is reserved and always has the value 0.
wr_dq0_del	Write dq0 delay fine-tuning. This field holds the number of delay units that are added to dq0 relative to dqs0. 00 No change in dq0 delay 01 Add dq0 delay of 1 delay unit. 10 Add dq0 delay of 2 delay units. 11 Add dq0 delay of 3 delay units.

44.12.43 MMDC PHY Write DQ Byte1 Delay Register (MMDCx_MPWRDQBY1DL)

This register is used to add fine-tuning adjustment to every bit in the write DQ byte1 relative to the write DQS

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: DDR3_x64, LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 830h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	wr_dm1_del		wr_dq15_del		0		wr_dq14_del		0		wr_dq13_del		0		wr_dq12_del	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0		wr_dq11_del		0		wr_dq10_del		0		wr_dq9_del		0		wr_dq8_del	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDCx_MPWRDQBY1DL field descriptions

Field	Description
31–30 wr_dm1_del	Write dm1 delay fine-tuning. This field holds the number of delay units that are added to dm1 relative to dqs1. 00 No change in dm1 delay 01 Add dm1 delay of 1 delay unit. 10 Add dm1 delay of 2 delay units. 11 Add dm1 delay of 3 delay units.
29–28 wr_dq15_del	Write dq15 delay fine-tuning. This field holds the number of delay units that are added to dq15 relative to dqs1. 00 No change in dq15 delay 01 Add dq15 delay of 1 delay unit. 10 Add dq15 delay of 2 delay units. 11 Add dq15 delay of 3 delay units.
27–26 Reserved	This read-only field is reserved and always has the value 0.
25–24 wr_dq14_del	Write dq14 delay fine-tuning. This field holds the number of delay units that are added to dq14 relative to dqs1. 00 No change in dq14 delay 01 Add dq14 delay of 1 delay unit. 10 Add dq14 delay of 2 delay units. 11 Add dq14 delay of 3 delay units.
23–22 Reserved	This read-only field is reserved and always has the value 0.
21–20 wr_dq13_del	Write dq13 delay fine-tuning. This field holds the number of delay units that are added to dq13 relative to dqs1. 00 No change in dq13 delay 01 Add dq13 delay of 1 delay unit. 10 Add dq13 delay of 2 delay units. 11 Add dq13 delay of 3 delay units.

Table continues on the next page...

MMDCx_MPWRDQBY1DL field descriptions (continued)

Field	Description
19–18 Reserved	This read-only field is reserved and always has the value 0.
17–16 wr_dq12_del	Write dq12 delay fine-tuning. This field holds the number of delay units that are added to dq12 relative to dqs1. 00 No change in dq12 delay 01 Add dq12 delay of 1 delay unit. 10 Add dq12 delay of 2 delay units. 11 Add dq12 delay of 3 delay units.
15–14 Reserved	This read-only field is reserved and always has the value 0.
13–12 wr_dq11_del	Write dq11 delay fine-tuning. This field holds the number of delay units that are added to dq11 relative to dqs1. 00 No change in dq11 delay 01 Add dq11 delay of 1 delay unit. 10 Add dq11 delay of 2 delay units. 11 Add dq11 delay of 3 delay units.
11–10 Reserved	This read-only field is reserved and always has the value 0.
9–8 wr_dq10_del	Write dq10 delay fine-tuning. This field holds the number of delay units that are added to dq10 relative to dqs1. 00 No change in dq10 delay 01 Add dq10 delay of 1 delay unit. 10 Add dq10 delay of 2 delay units. 11 Add dq10 delay of 3 delay units.
7–6 Reserved	This read-only field is reserved and always has the value 0.
5–4 wr_dq9_del	Write dq9 delay fine-tuning. This field holds the number of delay units that are added to dq9 relative to dqs1. 00 No change in dq9 delay 01 Add dq9 delay of 1 delay unit. 10 Add dq9 delay of 2 delay units. 11 Add dq9 delay of 3 delay units.
3–2 Reserved	This read-only field is reserved and always has the value 0.
wr_dq8_del	Write dq8 delay fine-tuning. This field holds the number of delay units that are added to dq8 relative to dqs1. 00 No change in dq8 delay 01 Add dq8 delay of 1 delay unit. 10 Add dq8 delay of 2 delay units. 11 Add dq8 delay of 3 delay units.

44.12.44 MMDC PHY Write DQ Byte2 Delay Register (MMDCx_MPWRDQBY2DL)

This register is used to add fine-tuning adjustment to every bit in the write DQ byte2 relative to the write DQS

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: DDR3_x64, LP2_2ch_x32

Address: Base address + 834h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	wr_dm2_del		wr_dq23_del		0		wr_dq22_del		0		wr_dq21_del		0		wr_dq20_del	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0		wr_dq19_del		0		wr_dq18_del		0		wr_dq17_del		0		wr_dq16_del	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDCx_MPWRDQBY2DL field descriptions

Field	Description
31–30 wr_dm2_del	Write dm2 delay fine-tuning. This field holds the number of delay units that are added to dm2 relative to dqs2. 00 No change in dm2 delay 01 Add dm2 delay of 1 delay unit. 10 Add dm2 delay of 2 delay units. 11 Add dm2 delay of 3 delay units.
29–28 wr_dq23_del	Write dq23 delay fine tuning. This field holds the number of delay units that are added to dq23 relative to dqs2. 00 No change in dq23 delay 01 Add dq23 delay of 1 delay unit. 10 Add dq23 delay of 2 delay units. 11 Add dq23 delay of 3 delay units.
27–26 Reserved	This read-only field is reserved and always has the value 0.
25–24 wr_dq22_del	Write dq22 delay fine tuning. This field holds the number of delay units that are added to dq22 relative to dqs2. 00 No change in dq22 delay 01 Add dq22 delay of 1 delay unit. 10 Add dq22 delay of 2 delay units. 11 Add dq22 delay of 3 delay units.

Table continues on the next page...

MMDCx_MPWRDQBY2DL field descriptions (continued)

Field	Description
23–22 Reserved	This read-only field is reserved and always has the value 0.
21–20 wr_dq21_del	Write dq21 delay fine tuning. This field holds the number of delay units that are added to dq21 relative to dqs2. 00 No change in dq21 delay 01 Add dq21 delay of 1 delay unit. 10 Add dq21 delay of 2 delay units. 11 Add dq21 delay of 3 delay units.
19–18 Reserved	This read-only field is reserved and always has the value 0.
17–16 wr_dq20_del	Write dq20 delay fine tuning. This field holds the number of delay units that are added to dq20 relative to dqs2. 00 No change in dq20 delay 01 Add dq20 delay of 1 delay unit. 10 Add dq20 delay of 2 delay units. 11 Add dq20 delay of 3 delay units.
15–14 Reserved	This read-only field is reserved and always has the value 0.
13–12 wr_dq19_del	Write dq19 delay fine tuning. This field holds the number of delay units that are added to dq19 relative to dqs2. 00 No change in dq19 delay 01 Add dq19 delay of 1 delay unit. 10 Add dq19 delay of 2 delay units. 11 Add dq19 delay of 3 delay units.
11–10 Reserved	This read-only field is reserved and always has the value 0.
9–8 wr_dq18_del	Write dq18 delay fine tuning. This field holds the number of delay units that are added to dq18 relative to dqs2. 00 No change in dq18 delay 01 Add dq18 delay of 1 delay unit. 10 Add dq18 delay of 2 delay units. 11 Add dq18 delay of 3 delay units.
7–6 Reserved	This read-only field is reserved and always has the value 0.
5–4 wr_dq17_del	Write dq17 delay fine tuning. This field holds the number of delay units that are added to dq17 relative to dqs2. 00 No change in dq17 delay 01 Add dq17 delay of 1 delay unit. 10 Add dq17 delay of 2 delay units. 11 Add dq17 delay of 3 delay units.
3–2 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

MMDCx_MPWRDQBY2DL field descriptions (continued)

Field	Description
wr_dq16_del	Write dq16 delay fine tuning. This field holds the number of delay units that are added to dq16 relative to dqs2. 00 No change in dq16 delay 01 Add dq16 delay of 1 delay unit. 10 Add dq16 delay of 2 delay units. 11 Add dq16 delay of 3 delay units.

44.12.45 MMDC PHY Write DQ Byte3 Delay Register (MMDCx_MPWRDQBY3DL)

This register is used to add fine-tuning adjustment to every bit in the write DQ byte3 relative to the write DQS

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: DDR3_x64, LP2_2ch_x32

Address: Base address + 838h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	wr_dm3_del		wr_dq31_del		0		wr_dq30_del		0		wr_dq29_del		0		wr_dq28_del	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0		wr_dq27_del		0		wr_dq26_del		0		wr_dq25_del		0		wr_dq24_del	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDCx_MPWRDQBY3DL field descriptions

Field	Description
31–30 wr_dm3_del	Write dm3 delay fine tuning. This field holds the number of delay units that are added to dm3 relative to dqs3. 00 No change in dm3 delay 01 Add dm3 delay of 1 delay unit. 10 Add dm3 delay of 2 delay units. 11 Add dm3 delay of 3 delay units.
29–28 wr_dq31_del	Write dq31 delay fine tuning. This field holds the number of delay units that are added to dq31 relative to dqs3. 00 No change in dq31 delay 01 Add dq31 delay of 1 delay unit.

Table continues on the next page...

MMDCx_MPWRDQBY3DL field descriptions (continued)

Field	Description
	10 Add dq31 delay of 2 delay units. 11 Add dq31 delay of 3 delay units.
27–26 Reserved	This read-only field is reserved and always has the value 0.
25–24 wr_dq30_del	Write dq30 delay fine tuning. This field holds the number of delay units that are added to dq30 relative to dqs3. 00 No change in dq30 delay 01 Add dq30 delay of 1 delay unit. 10 Add dq30 delay of 2 delay units. 11 Add dq30 delay of 3 delay units.
23–22 Reserved	This read-only field is reserved and always has the value 0.
21–20 wr_dq29_del	Write dq29 delay fine tuning. This field holds the number of delay units that are added to dq29 relative to dqs3. 00 No change in dq29 delay 01 Add dq29 delay of 1 delay unit. 10 Add dq29 delay of 2 delay units. 11 Add dq29 delay of 3 delay units.
19–18 Reserved	This read-only field is reserved and always has the value 0.
17–16 wr_dq28_del	Write dq28 delay fine tuning. This field holds the number of delay units that are added to dq28 relative to dqs3. 00 No change in dq28 delay 01 Add dq28 delay of 1 delay unit. 10 Add dq28 delay of 2 delay units. 11 Add dq28 delay of 3 delay units.
15–14 Reserved	This read-only field is reserved and always has the value 0.
13–12 wr_dq27_del	Write dq27 delay fine tuning. This field holds the number of delay units that are added to dq27 relative to dqs3. 00 No change in dq27 delay 01 Add dq27 delay of 1 delay unit. 10 Add dq27 delay of 2 delay units. 11 Add dq27 delay of 3 delay units.
11–10 Reserved	This read-only field is reserved and always has the value 0.
9–8 wr_dq26_del	Write dq26 delay fine tuning. This field holds the number of delay units that are added to dq26 relative to dqs3. 00 No change in dq26 delay 01 Add dq26 delay of 1 delay unit. 10 Add dq26 delay of 2 delay units. 11 Add dq26 delay of 3 delay units.

Table continues on the next page...

MMDCx_MPWRDQBY3DL field descriptions (continued)

Field	Description
7–6 Reserved	This read-only field is reserved and always has the value 0.
5–4 wr_dq25_del	Write dq25 delay fine tuning. This field holds the number of delay units that are added to dq25 relative to dqs3. 00 No change in dq25 delay 01 Add dq25 delay of 1 delay unit. 10 Add dq25 delay of 2 delay units. 11 Add dq25 delay of 3 delay units.
3–2 Reserved	This read-only field is reserved and always has the value 0.
wr_dq24_del	Write dq24 delay fine tuning. This field holds the number of delay units that are added to dq24 relative to dqs3. 00 No change in dq24 delay 01 Add dq24 delay of 1 delay unit. 10 Add dq24 delay of 2 delay units. 11 Add dq24 delay of 3 delay units.

44.12.46 MMDC PHY Read DQS Gating Control Register 0 (MMDCx_MPDGCTRL0)

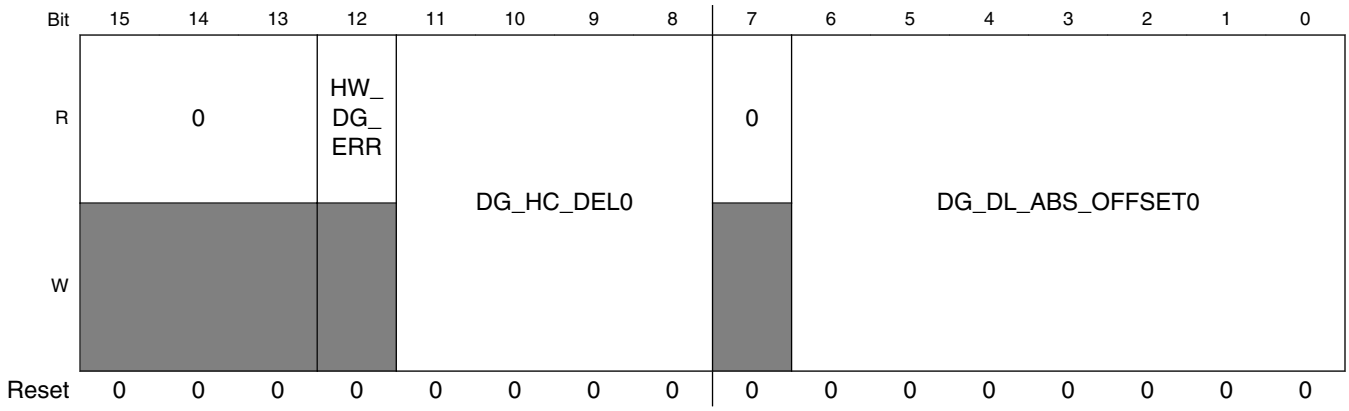
Supported Mode Of Operations:

For Channel 0: DDR3_x16, DDR3_x32, DDR3_x64

For Channel 1: DDR3_x64

Address: Base address + 83Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RST_RD_FIFO	DG_CMP_CYC	DG_DIS	HW_DG_EN	DG_HC_DEL1				DG_EXT_UP	DG_DL_ABS_OFFSET1						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



MMDCx_MPDGCTRL0 field descriptions

Field	Description
31 RST_RD_FIFO	Reset Read Data FIFO and associated pointers. If this bit is asserted then the MMDC resets the read data FIFO and the associated pointers. This bit is self cleared after the FIFO reset is done.
30 DG_CMP_CYC	Read DQS gating sample cycle. If this bit is asserted then the MMDC waits 32 cycles before comparing the read data, Otherwise it waits 16 DDR cycles. 0 MMDC waits 16 DDR cycles 1 MMDC waits 32 DDR cycles
29 DG_DIS	Read DQS gating disable. If this bit is asserted then the MMDC disables the read DQS gating mechnism. If this bits is asserted (read DQS gating is disabled) then pulll-up and pull-down resistors suppose to be used on DQS and DQS# respectively 0 Read DQS gating mechanism is enbled 1 Read DQS gating mechanism is disabled
28 HW_DG_EN	Enable automatic read DQS gating calibration. If this bit is asserted then the MMDC performs automatic read DQS gating calibration. HW negates this bit upon completion of the automatic read DQS gating. Note: Before issuing the first read command the MMDC counts 12 cycles. In LPDDR2 mode automatic (HW) read DQS gating should be disabled and Pull-up/pull-down resistors on DQS/DQS# should be enabled while ODT resistors must be disconnected. 0 Disable automatic read DQS gating calibration 1 Start automatic read DQS gating calibration
27–24 DG_HC_DEL1	Read DQS gating half cycles delay for Byte1 (channel 0 register) and Byte5 in 64-bit mode (channel 1 register) . This field indicates the delay in half cycles between read DQS gate and the middle of the read DQS preamble of Byte1. This delay is added to the delay that is genearted by the read DQS1 gating delay-line, So the total read DQS gating delay is (DG_HC_DEL#)*0.5*cycle + (DG_DL_ABS_OFFSET#)*1/256*cycle Upon completion of the automatic read DQS gating calibration this field gets the value of the 4 MSB of ((HW_DG_LOW1 + HW_DG_UP1) /2). 0000 0 cycles delay. 0001 Half cycle delay. 0010 1 cycle delay 1101 6.5 cycles delay

Table continues on the next page...

MMDCx_MPDGCTRL0 field descriptions (continued)

Field	Description
	1110 Reserved 1111 Reserved
23 DG_EXT_UP	DG extend upper boundary. By default the upper boundary of DQS gating HW calibration is set according to first failing comparison after at least one passing comparison. If this bit is asserted then the upper boundary is set accroding to the last passing comparison.
22–16 DG_DL_ABS_OFFSET1	Absolute read DQS gating delay offset for Byte1. This field indicates the absolute delay between read DQS gate and the middle of the read DQS preamble of Byte1 with fractions of a clock period and up to half cycle.The fraction is process and frequency independent. The delay of the delay-line would be (DG_DL_ABS_OFFSET1 / 256)* MMDC_CH0 AXI clock (fast clock). This field can also bit written by HW. Upon completion of the automatic read DQS gating calibration this field gets the value of the 7 LSB of ((HW_DG_LOW1 + HW_DG_UP1) /2). Note that not all changes will have effect on the actual delay. If the requested change is smaller than the delay-line resolution, then no change will occur.
15–13 Reserved	This read-only field is reserved and always has the value 0.
12 HW_DG_ERR	HW DQS gating error. This bit valid is asserted when an error was found during the read DQS gating HW calibration process. Error can occur when no valid value was found during HW calibration. This bit is valid only after HW_DG_EN is de-asserted. 0 No error was found during the DQS gating HW calibration process. 1 An error was found during the DQS gating HW calibration process.
11–8 DG_HC_DEL0	Read DQS gating half cycles delay for Byte0 (Channel 0 register) and Byte4 in 64-bit mode (Channel 1 register) . This field indicates the delay in half cycles between read DQS gate and the middle of the read DQS preamble of Byte0/4. This delay is added to the delay that is genearted by the read DQS1 gating delay-line, So the total read DQS gating delay is (DG_HC_DEL#)*0.5*cycle + (DG_DL_ABS_OFFSET#)*1/256*cycle Upon completion of the automatic read DQS gating calibration this field gets the value of the 4 MSB of ((HW_DG_LOW1 + HW_DG_UP1) /2). 0000 0 cycles delay. 0001 Half cycle delay. 0010 1 cycle delay 1101 6.5 cycles delay 1110 Reserved 1111 Reserved
7 Reserved	This read-only field is reserved and always has the value 0.
DG_DL_ABS_OFFSET0	Absolute read DQS gating delay offset for Byte0. This field indicates the absolute delay between read DQS gate and the middle of the read DQS preamble of Byte0 with fractions of a clock period and up to half cycle.The fraction is process and frequency independent. The delay of the delay-line would be (DG_DL_ABS_OFFSET0 / 256)* MMDC_CH0 AXI clock (fast clock). This field can also bit written by HW. Upon completion of the automatic read DQS gating calibration this field gets the value of the 7 LSB of ((HW_DG_LOW0 + HW_DG_UP0) /2). Note that not all changes will have effect on the actual delay. If the requested change is smaller than the delay-line resolution, then no change will occur.

44.12.47 MMDC PHY Read DQS Gating Control Register 1 (MMDCx_MPDGCTRL1)

Supported Mode Of Operations:

For Channel 0: DDR3_x16, DDR3_x32, DDR3_x64

For Channel 1: DDR3_x64

Address: Base address + 840h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0				DG_HC_DEL3				0	DG_DL_ABS_OFFSET3						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				DG_HC_DEL2				0	DG_DL_ABS_OFFSET2						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDCx_MPDGCTRL1 field descriptions

Field	Description
31–28 Reserved	This read-only field is reserved and always has the value 0.
27–24 DG_HC_DEL3	Read DQS gating half cycles delay for Byte3 (Channel 0 register) and Byte7 for 64-bit data (Channel 1 register) . This field indicates the delay in half cycles between read DQS gate and the middle of the read DQS preamble of Byte3/7. This delay is added to the delay that is genearted by the read DQS1 gating delay-line, So the total read DQS gating delay is (DG_HC_DEL#)*0.5*cycle + (DG_DL_ABS_OFFSET#)*1/256*cycle Upon completion of the automatic read DQS gating calibration this field gets the value of the 4 MSB of ((HW_DG_LOW3 + HW_DG_UP3) /2). 0000 0 cycles delay. 0001 Half cycle delay. 0010 1 cycle delay 1101 6.5 cycles delay 1110 Reserved 1111 Reserved
23 Reserved	This read-only field is reserved and always has the value 0.
22–16 DG_DL_ABS_ OFFSET3	Absolute read DQS gating delay offset for Byte3. This field indicates the absolute delay between read DQS gate and the middle of the read DQS preamble of Byte3 with fractions of a clock period and up to half cycle.The fraction is process and frequency independent. The delay of the delay-line would be (DG_DL_ABS_OFFSET3 / 256)* MMDC_CH0 AXI clock (fast clock). This field can also bit written by HW. Upon completion of the automatic read DQS gating calibration this field gets the value of the 7 LSB of ((HW_DG_LOW3 + HW_DG_UP3) /2).

Table continues on the next page...

MMDCx_MPDGCTRL1 field descriptions (continued)

Field	Description
	Note that not all changes will have effect on the actual delay. If the requested change is smaller than the delay-line resolution, then no change will occur.
15–12 Reserved	This read-only field is reserved and always has the value 0.
11–8 DG_HC_DEL2	Read DQS gating half cycles delay for Byte2 (Channel 0 register) and Byte6 for 64-bit mode(channel 1 register) . This field indicates the delay in half cycles between read DQS gate and the middle of the read DQS preamble of Byte2/5. This delay is added to the delay that is genearted by the read DQS1 gating delay-line, So the total read DQS gating delay is (DG_HC_DEL#)*0.5*cycle + (DG_DL_ABS_OFFSET#)*1/256*cycle Upon completion of the automatic read DQS gating calibration this field gets the value of the 4 MSB of ((HW_DG_LOW2 + HW_DG_UP2) /2). 0000 0 cycles delay. 0001 Half cycle delay. 0010 1 cycle delay 1101 6.5 cycles delay 1110 Reserved 1111 Reserved
7 Reserved	This read-only field is reserved and always has the value 0.
DG_DL_ABS_ OFFSET2	Absolute read DQS gating delay offset for Byte2. This field indicates the absolute delay between read DQS gate and the middle of the read DQS preamble of Byte2 with fractions of a clock period and up to half cycle.The fraction is process and frequency independent. The delay of the delay-line would be (DG_DL_ABS_OFFSET2 / 256)* MMDC_CH0 AXI clock (fast clock). This field can also bit written by HW. Upon completion of the automatic read DQS gating calibration this field gets the value of the 7 LSB of ((HW_DG_LOW2 + HW_DG_UP2) /2). Note that not all changes will have effect on the actual delay. If the requested change is smaller than the delay-line resolution, then no change will occur.

44.12.48 MMDC PHY Read DQS Gating delay-line Status Register (MMDCx_MPDGDLST0)

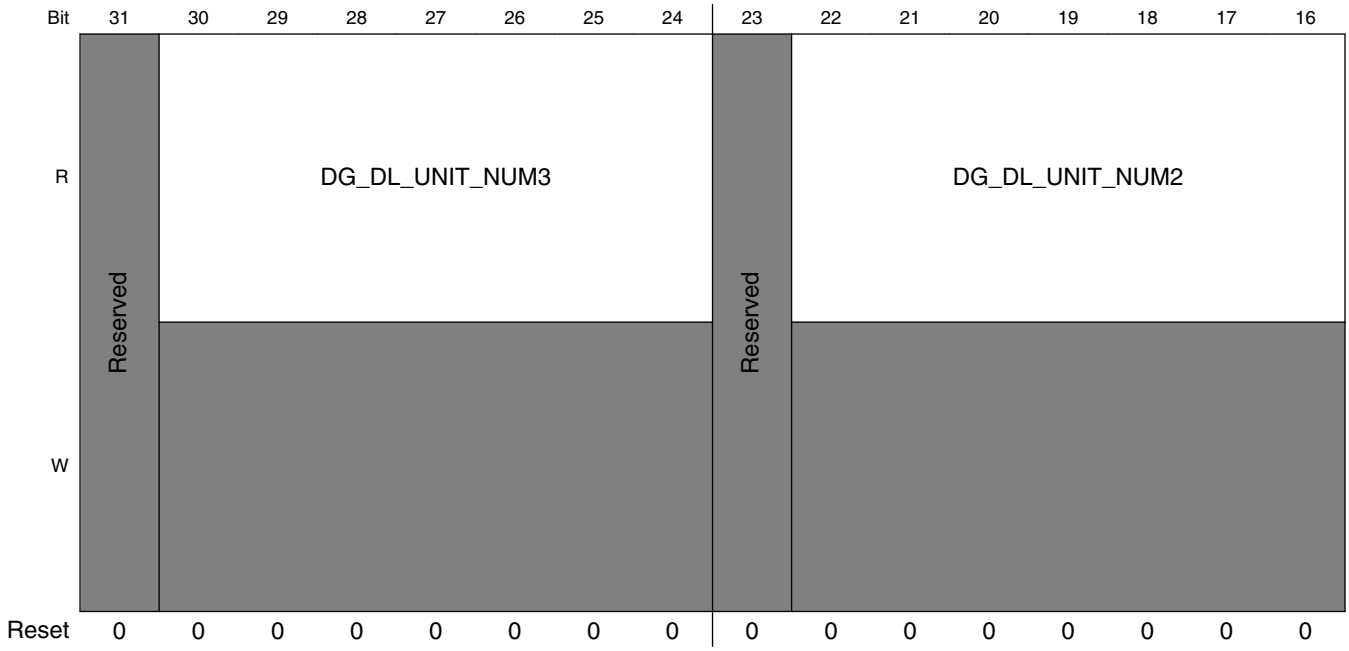
This register holds the status of the 4 dqs gating delay-lines.

Supported Mode Of Operations:

For Channel 0: DDR3_x16, DDR3_x32, DDR3_x64

For Channel 1: DDR3_x64

Address: Base address + 844h offset



MMDC*x*_MPDGDLST0 field descriptions

Field	Description
31 -	This field is reserved. Reserved
30–24 DG_DL_UNIT_NUM3	This field reflects the number of delay units that are actually used by read DQS gating delay-line 3.
23 -	This field is reserved. Reserved

Table continues on the next page...

MMDCx_MPDGDLST0 field descriptions (continued)

Field	Description
22–16 DG_DL_UNIT_NUM2	This field reflects the number of delay units that are actually used by read DQS gating delay-line 2.
15 -	This field is reserved. Reserved
14–8 DG_DL_UNIT_NUM1	This field reflects the number of delay units that are actually used by read DQS gating delay-line 1.
7 -	This field is reserved. Reserved
DG_DL_UNIT_NUM0	This field reflects the number of delay units that are actually used by read DQS gating delay-line 0.

44.12.49 MMDC PHY Read delay-lines Configuration Register (MMDCx_MPRDDLCTL)

This register controls read delay-lines functionality; it determines DQS delay relative to the associated DQ read access. The delay-line compensates for process variations and produces a constant delay regardless of the process, temperature and voltage.

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: DDR3_x64, LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 848h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	RD_DL_ABS_OFFSET3							0	RD_DL_ABS_OFFSET2						
W																
Reset	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	RD_DL_ABS_OFFSET1							0	RD_DL_ABS_OFFSET0						
W																
Reset	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0

MMDCx_MPRDDLCTL field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30–24 RD_DL_ABS_OFFSET3	Absolute read delay offset for Byte3. This field indicates the absolute delay between read DQS strobe and the read data of Byte3 with fractions of a clock period and up to half cycle. The fraction is process and

Table continues on the next page...

MMDCx_MPRDDLCTL field descriptions (continued)

Field	Description
	frequency independent. The delay of the delay-line would be (RD_DL_ABS_OFFSET3 / 256) * MMDC_CH0 AXI clock (fast clock). So for the default value of 64 we get a quarter cycle delay. This field can also bit written by HW. Upon completion of the read delay-line HW calibration this field gets the value of (HW_RD_DL_LOW3 + HW_RD_DL_UP3) /2 Note that not all changes will have effect on the actual delay. If the requested change is smaller than the delay-line resolution, then no change will occur.
23 Reserved	This read-only field is reserved and always has the value 0.
22–16 RD_DL_ABS_OFFSET2	Absolute read delay offset for Byte2. This field indicates the absolute delay between read DQS strobe and the read data of Byte2 with fractions of a clock period and up to half cycle. The fraction is process and frequency independent. The delay of the delay-line would be (RD_DL_ABS_OFFSET2 / 256) * MMDC_CH0 AXI clock (fast clock). So for the default value of 64 we get a quarter cycle delay. This field can also bit written by HW. Upon completion of the read delay-line HW calibration this field gets the value of (HW_RD_DL_LOW2 + HW_RD_DL_UP2) /2 Note that not all changes will have effect on the actual delay. If the requested change is smaller than the delay-line resolution, then no change will occur.
15 Reserved	This read-only field is reserved and always has the value 0.
14–8 RD_DL_ABS_OFFSET1	Absolute read delay offset for Byte1. This field indicates the absolute delay between read DQS strobe and the read data of Byte1 with fractions of a clock period and up to half cycle. The fraction is process and frequency independent. The delay of the delay-line would be (RD_DL_ABS_OFFSET1 / 256) * MMDC_CH0 AXI clock (fast clock). So for the default value of 64 we get a quarter cycle delay. This field can also bit written by HW. Upon completion of the read delay-line HW calibration this field gets the value of (HW_RD_DL_LOW1 + HW_RD_DL_UP1) /2 Note that not all changes will have effect on the actual delay. If the requested change is smaller than the delay-line resolution, then no change will occur.
7 Reserved	This read-only field is reserved and always has the value 0.
RD_DL_ABS_OFFSET0	Absolute read delay offset for Byte0. This field indicates the absolute delay between read DQS strobe and the read data of Byte0 with fractions of a clock period and up to half cycle. The fraction is process and frequency independent. The delay of the delay-line would be (RD_DL_ABS_OFFSET0 / 256) * MMDC_CH0 AXI clock (fast clock). So for the default value of 64 we get a quarter cycle delay. This field can also bit written by HW. Upon completion of the read delay-line HW calibration this field gets the value of (HW_RD_DL_LOW0 + HW_RD_DL_UP0) /2 Note that not all changes will have effect on the actual delay. If the requested change is smaller than the delay-line resolution, then no change will occur.

44.12.50 MMDC PHY Read delay-lines Status Register (MMDCx_MPRDDLST)

This register holds the status of the 4 read delay-lines.

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: DDR3_x64, LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 84Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	RD_DL_UNIT_NUM3							0	RD_DL_UNIT_NUM2						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	RD_DL_UNIT_NUM1							0	RD_DL_UNIT_NUM0						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDCx_MPRDDLST field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30–24 RD_DL_UNIT_NUM3	This field reflects the number of delay units that are actually used by read delay-line 3.
23 Reserved	This read-only field is reserved and always has the value 0.
22–16 RD_DL_UNIT_NUM2	This field reflects the number of delay units that are actually used by read delay-line 2.
15 Reserved	This read-only field is reserved and always has the value 0.
14–8 RD_DL_UNIT_NUM1	This field reflects the number of delay units that are actually used by read delay-line 1.
7 Reserved	This read-only field is reserved and always has the value 0.
RD_DL_UNIT_NUM0	This field reflects the number of delay units that are actually used by read delay-line 0.

44.12.51 MMDC PHY Write delay-lines Configuration Register (MMDCx_MPWRDLCTL)

This register controls write delay-lines functionality, it determines DQ/DM delay relative to the associated DQS in write access. The delay-line compensates for process variations, and produces a constant delay regardless of the process, temperature and voltage.

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: DDR3_x64, LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 850h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WR_DL_ABS_OFFSET3							0	WR_DL_ABS_OFFSET2						
W																
Reset	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WR_DL_ABS_OFFSET1							0	WR_DL_ABS_OFFSET0						
W																
Reset	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0

MMDCx_MPWRDLCTL field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30–24 WR_DL_ABS_ OFFSET3	Absolute write delay offset for Byte3. This field indicates the absolute delay between write DQS strobe and the write data of Byte3 with fractions of a clock period and up to half cycle. The fraction is process and frequency independent. The delay of the delay-line would be (WR_DL_ABS_OFFSET3 / 256) * MMDC_CH0 AXI clock (fast clock). So for the default value of 64 we get a quarter cycle delay. This field can also bit written by HW. Upon completion of the write delay-line HW calibration this field gets the value of (HW_WR_DL_LOW3 + HW_WR_DL_UP3) /2 Note that not all changes will have effect on the actual delay. If the requested change is smaller than the delay-line resolution, then no change will occur.
23 Reserved	This read-only field is reserved and always has the value 0.
22–16 WR_DL_ABS_ OFFSET2	Absolute write delay offset for Byte2. This field indicates the absolute delay between write DQS strobe and the write data of Byte2 with fractions of a clock period and up to half cycle. The fraction is process and frequency independent. The delay of the delay-line would be (WR_DL_ABS_OFFSET2/ 256) * MMDC_CH0 AXI clock (fast clock). So for the default value of 64 we get a quarter cycle delay. This field can also bit written by HW. Upon completion of the write delay-line HW calibration this field gets the value of (HW_WR_DL_LOW2 + HW_WR_DL_UP2) /2 Note that not all changes will have effect on the actual delay. If the requested change is smaller than the delay-line resolution, then no change will occur.
15 Reserved	This read-only field is reserved and always has the value 0.
14–8 WR_DL_ABS_ OFFSET1	Absolute write delay offset for Byte1. This field indicates the absolute delay between write DQS strobe and the write data of Byte1 with fractions of a clock period and up to half cycle. The fraction is process and frequency independent. The delay of the delay-line would be (WR_DL_ABS_OFFSET1 / 256) * MMDC_CH0 AXI clock (fast clock). So for the default value of 64 we get a quarter cycle delay. This field can also bit written by HW. Upon completion of the write delay-line HW calibration this field gets the value of (HW_WR_DL_LOW1 + HW_WR_DL_UP1) /2 Note that not all changes of this value will affect the actual delay. If the requested change is smaller than the delay-line resolution, then no change will occur.
7 Reserved	This read-only field is reserved and always has the value 0.
WR_DL_ABS_ OFFSET0	Absolute write delay offset for Byte0. This field indicates the absolute delay between write DQS strobe and the write data of Byte3 with fractions of a clock period and up to half cycle. The fraction is process and

Table continues on the next page...

MMDCx_MPWRDLCTL field descriptions (continued)

Field	Description
	frequency independent. The delay of the delay-line would be (WR_DL_ABS_OFFSET0 / 256) * MMDC_CH0 AXI clock (fast clock). So for the default value of 64 we get a quarter cycle delay. This field can also bit written by HW. Upon completion of the write delay-line HW calibration this field gets the value of (HW_WR_DL_LOW0 + HW_WR_DL_UP0) /2 Note that not all changes of this value will affect the actual delay. If the requested change is smaller than the delay-line resolution, then no change will occur.

44.12.52 MMDC PHY Write delay-lines Status Register (MMDCx_MPWRDLST)

This register holds the status of the 4 write delay-line.

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: DDR3_x64, LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 854h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	WR_DL_UNIT_NUM3							0	WR_DL_UNIT_NUM2						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WR_DL_UNIT_NUM1							0	WR_DL_UNIT_NUM0						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDCx_MPWRDLST field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30–24 WR_DL_UNIT_NUM3	This field reflects the number of delay units that are actually used by write delay-line 3.
23 Reserved	This read-only field is reserved and always has the value 0.
22–16 WR_DL_UNIT_NUM2	This field reflects the number of delay units that are actually used by write delay-line 2.
15 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

MMDCx_MPWRDLST field descriptions (continued)

Field	Description
14–8 WR_DL_UNIT_NUM1	This field reflects the number of delay units that are actually used by write delay-line 1.
7 Reserved	This read-only field is reserved and always has the value 0.
WR_DL_UNIT_NUM0	This field reflects the number of delay units that are actually used by write delay-line 0.

44.12.53 MMDC PHY CK Control Register (MMDCx_MPSDCTRL)

This register controls the fine tuning of the primary clock (CK0).

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 858h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0						SDclk0_del		0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDCx_MPSDCTRL field descriptions

Field	Description
31–10 Reserved	This read-only field is reserved and always has the value 0.
9–8 SDclk0_del	DDR clock0 delay fine tuning. This field holds the number of delay units that are added to DDR clock (CK0). Note: In case of LPDDR2 2-ch mode this registers controls the fine tuning of the clock that is driven to channel0 In case of DDR3 the fine tuning of the secondary clock is controlled by 0x021B_4858[SDCLK] 00 No change in DDR clock0 delay 01 Add DDR clock0 delay of 1 delay unit. 10 Add DDR clock0 delay of 2 delay units. 11 Add DDR clock0 delay of 3 delay units.
Reserved	This read-only field is reserved and always has the value 0.

44.12.54 MMDC ZQ LPDDR2 HW Control Register (MMDCx_MPZQLP2CTL)

This register controls the idle time that takes the LPDDR2 device to perform ZQ calibration

Supported Mode Of Operations:

For Channel 0: LP2_2ch_x16, LP2_2ch_x32

For Channel 1: LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 85Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	ZQ_LP2_HW_ZQCS							ZQ_LP2_HW_ZQCL							
W																
Reset	0	0	0	0	1	1	0	1	1	0	1	0	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								ZQ_LP2_HW_ZQINIT							
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1

MMDCx_MPZQLP2CTL field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30–24 ZQ_LP2_HW_ZQCS	This register defines the period in cycles that it takes the memory device to perform a Short ZQ calibration. This is the period of time that the MMDC has to wait after sending a long ZQ calibration and before sending other commands. This delay will also be used if ZQ reset is sent. 0x0-0x1A Reserved 0x1B 112 cycles (default) 0x1C 116 cycles 0x7E 508 cycles 0x7F 512 cycles
23–16 ZQ_LP2_HW_ZQCL	This register defines the period in cycles that it takes the memory device to perform a long ZQ calibration. This is the period of time that the MMDC has to wait after sending a Short ZQ calibration and before sending other commands. 0x0-0x36 Reserved 0x37 112 cycles 0x38 114 cycles 0x5F 192 cycles (Default, JEDEC value, tZQCL, for LPDDR2, 360ns @ clock frequency 533MHz) 0xFE 510 cycles 0xFF 512 cycles

Table continues on the next page...

MMDCx_MPZQLP2CTL field descriptions (continued)

Field	Description
15–9 Reserved	This read-only field is reserved and always has the value 0.
ZQ_LP2_HW_ ZQINIT	This register defines the period in cycles that it takes the memory device to perform a Init ZQ calibration. This is the period of time that the MMDC has to wait after sending a init ZQ calibration and before sending other commands. 0x0-0x36 Reserved 0x37 112 cycles 0x38 114 cycles 0x109 532 cycles (Default, JEDEC value, tZQINIT, for LPDDR2, 1us @ clock frequency 533MHz) 0x1FE 1022 cycles 0x1FF 1024 cycles

44.12.55 MMDC PHY Read Delay HW Calibration Control Register (MMDCx_MPRDDLHWCTL)

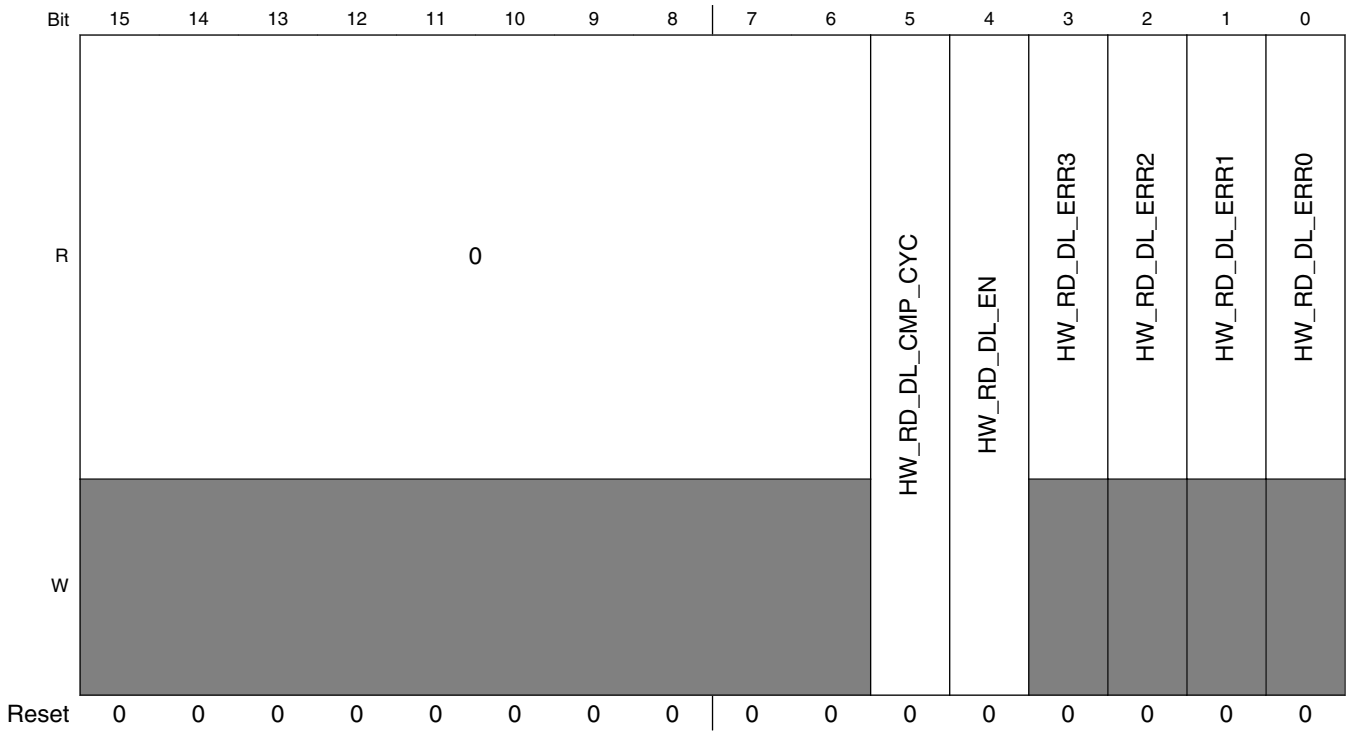
Supported Mode Of Operations:

For Channel 0: All

For Channel 1: DDR3_x64, LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 860h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



MMDCx_MPRDDLHWCTL field descriptions

Field	Description
31–6 Reserved	This read-only field is reserved and always has the value 0.
5 HW_RD_DL_ CMP_CYC	Automatic (HW) read sample cycle. If this bit is asserted then the MMDC will compare the read data 32 cycles after the MMDC sent the read command enable pulse else it compares the data after 16 cycles.
4 HW_RD_DL_EN	Enable automatic (HW) read calibration. If this bit is asserted then the MMDC will perform an automatic read calibration. HW should negate this bit upon completion of the calibration. Negation of this bit also points that the read calibration results are valid Note: Before issuing the first read command MMDC counts 12 cycles.
3 HW_RD_DL_ ERR3	Automatic (HW) read calibration error of Byte3. If this bit is asserted then it indicates that an error was found during the HW calibration process of read delay-line 3. In case this bit is zero at the end of the calibration process then the boundary results can be found at MPRDDLHWST1 register. This bit is valid only after HW_RD_DL_EN is de-asserted. 0 No error was found in read delay-line 3 during the automatic (HW) read calibration process of read delay-line 3. 1 An error was found in read delay-line 3 during the automatic (HW) read calibration process of read delay-line 3.
2 HW_RD_DL_ ERR2	Automatic (HW) read calibration error of Byte2. If this bit is asserted then it indicates that an error was found during the HW calibration process of read delay-line 2. In case this bit is zero at the end of the calibration process then the boundary results can be found at MPRDDLHWST1 register. This bit is valid only after HW_RD_DL_EN is de-asserted.

Table continues on the next page...

MMDCx_MPRDDLHWCTL field descriptions (continued)

Field	Description
	0 No error was found in read delay-line 2 during the automatic (HW) read calibration process of read delay-line 2. 1 An error was found in read delay-line 2 during the automatic (HW) read calibration process of read delay-line 2.
1 HW_RD_DL_ERR1	Automatic (HW) read calibration error of Byte1. If this bit is asserted then it indicates that an error was found during the HW calibration process of read delay-line 1. In case this bit is zero at the end of the calibration process then the boundary results can be found at MPRDDLHWST0 register. This bit is valid only after HW_RD_DL_EN is de-asserted. 0 No error was found in read delay-line 1 during the automatic (HW) read calibration process of read delay-line 1. 1 An error was found in read delay-line 1 during the automatic (HW) read calibration process of read delay-line 1.
0 HW_RD_DL_ERR0	Automatic (HW) read calibration error of Byte0. If this bit is asserted then it indicates that an error was found during the HW calibration process of read delay-line 0. In case this bit is zero at the end of the calibration process then the boundary results can be found at MPRDDLHWST0 register. This bit is valid only after HW_RD_DL_EN is de-asserted. 0 No error was found in read delay-line 0 during the automatic (HW) read calibration process of read delay-line 0. 1 An error was found in read delay-line 0 during the automatic (HW) read calibration process of read delay-line 0.

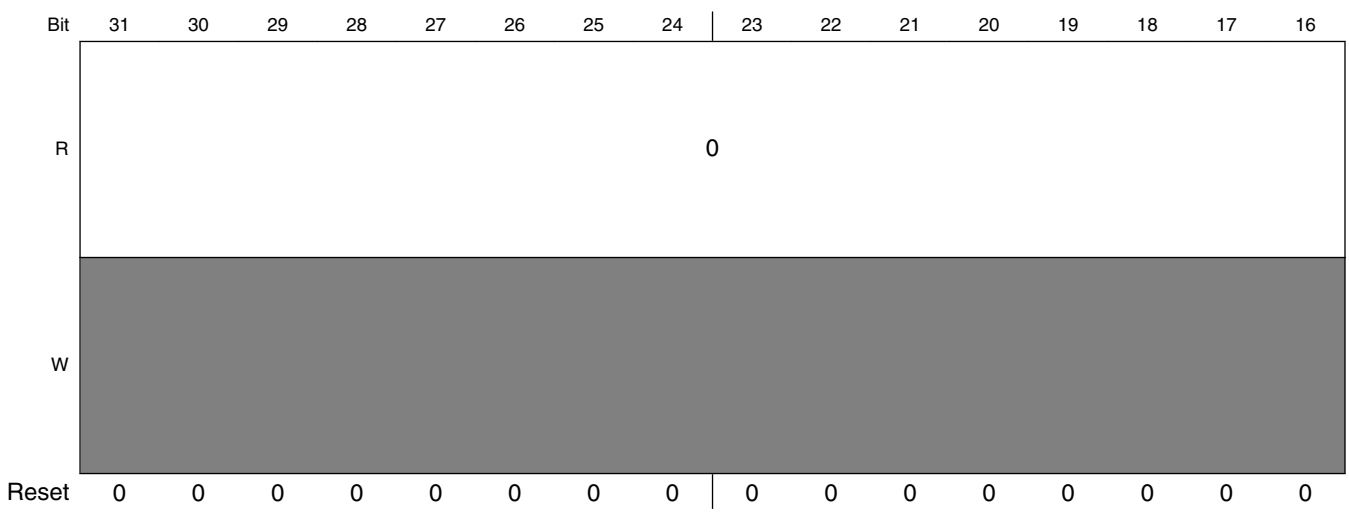
44.12.56 MMDC PHY Write Delay HW Calibration Control Register (MMDCx_MPWRDLHWCTL)

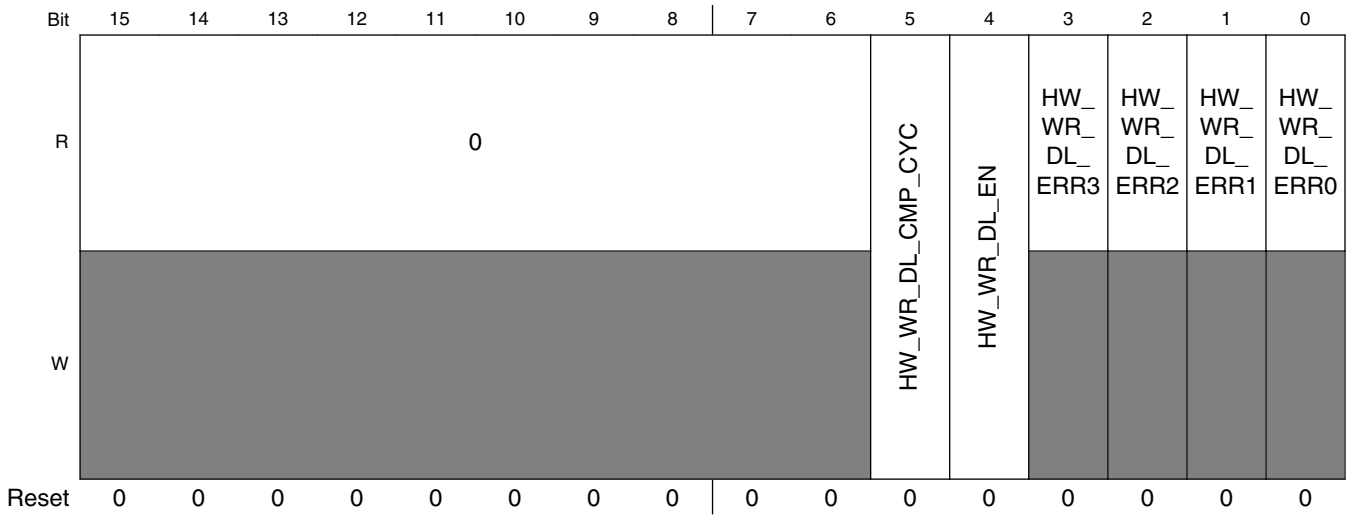
Supported Mode Of Operations:

For Channel 0: All

For Channel 1: DDR3_x64, LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 864h offset





MMDCx_MPWRDLHWCTL field descriptions

Field	Description
31–6 Reserved	This read-only field is reserved and always has the value 0.
5 HW_WR_DL_CMP_CYC	Write sample cycle. If this bit is asserted then the MMDC will compare the data 32 cycles after the MMDC sent the read command enable pulse else it compares the data after 16 cycles.
4 HW_WR_DL_EN	Enable automatic (HW) write calibration. If this bit is asserted then the MMDC will perform an automatic write calibration. HW should negate this bit upon completion of the calibration. Negation of this bit also indicates that the write calibration results are valid Note: Before issuing the first read command MMDC counts 12 cycles.
3 HW_WR_DL_ERR3	Automatic (HW) write calibration error of Byte3. If this bit is asserted then it indicates that an error was found during the HW calibration process of write delay-line 3. In case this bit is zero at the end of the calibration process then the boundary results can be found at MPWRDLHWST1 register. This bit is valid only after HW_WR_DL_EN is de-asserted. 0 No error was found during the automatic (HW) write calibration process of write delay-line 3. 1 An error was found during the automatic (HW) write calibration process of write delay-line 3.
2 HW_WR_DL_ERR2	Automatic (HW) write calibration error of Byte2. If this bit is asserted then it indicates that an error was found during the HW calibration process of write delay-line 2. T In case this bit is zero at the end of the calibration process then the boundary results can be found at MPWRDLHWST1 register. This bit is valid only after HW_WR_DL_EN is de-asserted. 0 No error was found during the automatic (HW) write calibration process of write delay-line 2. 1 An error was found during the automatic (HW) write calibration process of write delay-line 2.
1 HW_WR_DL_ERR1	Automatic (HW) write calibration error of Byte1. If this bit is asserted then it indicates that an error was found during the HW calibration process of write delay-line 1. In case this bit is zero at the end of the calibration process then the boundary results can be found at MPWRDLHWST0 register. This bit is valid only after HW_WR_DL_EN is de-asserted. 0 No error was found during the automatic (HW) write calibration process of write delay-line 1. 1 An error was found during the automatic (HW) write calibration process of write delay-line 1.
0 HW_WR_DL_ERR0	Automatic (HW) write calibration error of Byte0. If this bit is asserted then it indicates that an error was found during the HW calibration process of write delay-line 0. In case this bit is zero at the end of the

Table continues on the next page...

MMDCx_MPWRDLHWCTL field descriptions (continued)

Field	Description
	calibration process then the boundary results can be found at MPWRDLHWST0 register. This bit is valid only after HW_WR_DL_EN is de-asserted. 0 No error was found during the automatic (HW) write calibration process of write delay-line 0. 1 An error was found during the automatic (HW) write calibration process of write delay-line 0.

44.12.57 MMDC PHY Read Delay HW Calibration Status Register 0 (MMDCx_MPRDDLHWST0)

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: DDR3_x64, LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 868h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	HW_RD_DL_UP1							0	HW_RD_DL_LOW1						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	HW_RD_DL_UP0							0	HW_RD_DL_LOW0						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDCx_MPRDDLHWST0 field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30–24 HW_RD_DL_ UP1	Automatic (HW) read calibration result of the upper boundary of Byte1. This field holds the automatic (HW) read calibration result of the upper boundary of Byte1
23 Reserved	This read-only field is reserved and always has the value 0.
22–16 HW_RD_DL_ LOW1	Automatic (HW) read calibration result of the lower boundary of Byte1. This field holds the automatic (HW) read calibration result of the lower boundary of Byte1
15 Reserved	This read-only field is reserved and always has the value 0.
14–8 HW_RD_DL_ UP0	Automatic (HW) read calibration result of the upper boundary of Byte0. This field holds the automatic (HW) read calibration result of the upper boundary of Byte0.

Table continues on the next page...

MMDCx_MPRDDLHWST0 field descriptions (continued)

Field	Description
7 Reserved	This read-only field is reserved and always has the value 0.
HW_RD_DL_ LOW0	Automatic (HW) read calibration result of the lower boundary of Byte0. This field holds the automatic (HW) read calibration result of the lower boundary of Byte0.

44.12.58 MMDC PHY Read Delay HW Calibration Status Register 1 (MMDCx_MPRDDLHWST1)

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: DDR3_x64, LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 86Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	HW_RD_DL_UP3							0	HW_RD_DL_LOW3						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	HW_RD_DL_UP2							0	HW_RD_DL_LOW2						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDCx_MPRDDLHWST1 field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30–24 HW_RD_DL_ UP3	Automatic (HW) read calibration result of the upper boundary of Byte3. This field holds the automatic (HW) read calibration result of the upper boundary of Byte3
23 Reserved	This read-only field is reserved and always has the value 0.
22–16 HW_RD_DL_ LOW3	Automatic (HW) read calibration result of the lower boundary of Byte3. This field holds the automatic (HW) read calibration result of the lower boundary of Byte3
15 Reserved	This read-only field is reserved and always has the value 0.
14–8 HW_RD_DL_ UP2	Automatic (HW) read calibration result of the upper boundary of Byte2. This field holds the automatic (HW) read calibration result of the upper boundary of Byte2.

Table continues on the next page...

MMDCx_MPRDDLHWST1 field descriptions (continued)

Field	Description
7 Reserved	This read-only field is reserved and always has the value 0.
HW_RD_DL_ LOW2	Automatic (HW) read calibration result of the lower boundary of Byte2. This field holds the automatic (HW) read calibration result of the lower boundary of Byte2.

44.12.59 MMDC PHY Write Delay HW Calibration Status Register 0 (MMDCx_MPWRDLHWST0)

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: DDR3_x64, LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 870h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	HW_WR_DL_UP1							0	HW_WR_DL_LOW1						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	HW_WR_DL_UP0							0	HW_WR_DL_LOW0						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDCx_MPWRDLHWST0 field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30–24 HW_WR_DL_ UP1	Aautomatic (HW) write utomatic (HW) write calibration result of the upper boundary of Byte1. This field holds the automatic (HW) write calibration result of the upper boundary of Byte1.
23 Reserved	This read-only field is reserved and always has the value 0.
22–16 HW_WR_DL_ LOW1	Automatic (HW) write calibration result of the lower boundary of Byte1. This field holds the automatic (HW) write calibration result of the lower boundary of Byte1.
15 Reserved	This read-only field is reserved and always has the value 0.
14–8 HW_WR_DL_ UP0	Automatic (HW) write calibration result of the upper boundary of Byte0. This field holds the automatic (HW) write calibration result of the upper boundary of Byte0.

Table continues on the next page...

MMDCx_MPWRDLHWST0 field descriptions (continued)

Field	Description
7 Reserved	This read-only field is reserved and always has the value 0.
HW_WR_DL_ LOW0	Automatic (HW) write calibration result of the lower boundary of Byte0. This field holds the automatic (HW) write calibration result of the lower boundary of Byte0.

44.12.60 MMDC PHY Write Delay HW Calibration Status Register 1 (MMDCx_MPWRDLHWST1)

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: DDR3_x64, LP2_2ch_x32

Address: Base address + 874h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	HW_WR_DL_UP3							0	HW_WR_DL_LOW3						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	HW_WR_DL_UP2							0	HW_WR_DL_LOW2						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDCx_MPWRDLHWST1 field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30–24 HW_WR_DL_ UP3	Automatic (HW) write calibration result of the upper boundary of Byte3. This field holds the automatic (HW) write calibration result of the upper boundary of Byte3.
23 Reserved	This read-only field is reserved and always has the value 0.
22–16 HW_WR_DL_ LOW3	Automatic (HW) write calibration result of the lower boundary of Byte3. This field holds the automatic (HW) write calibration result of the lower boundary of Byte3.
15 Reserved	This read-only field is reserved and always has the value 0.
14–8 HW_WR_DL_ UP2	Automatic (HW) write calibration result of the upper boundary of Byte2. This field holds the automatic (HW) write calibration result of the upper boundary of Byte2.

Table continues on the next page...

MMDCx_MPWRDLHWST1 field descriptions (continued)

Field	Description
7 Reserved	This read-only field is reserved and always has the value 0.
HW_WR_DL_ LOW2	Automatic (HW) write calibration result of the lower boundary of Byte2. This field holds the automatic (HW) write calibration result of the lower boundary of Byte2.

44.12.61 MMDC PHY Write Leveling HW Error Register (MMDCx_MPWLHWERR)

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: DDR3_x64, LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 878h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HW_WL3_DQ								HW_WL2_DQ								HW_WL1_DQ								HW_WL0_DQ							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDCx_MPWLHWERR field descriptions

Field	Description
31–24 HW_WL3_DQ	HW write-leveling calibration result of Byte3. This field holds the results for all the 8 write-leveling steps of Byte3. i.e bit 0 holds the result of the write-leveling calibration of 0 delay, bit 1holds the result of the write-leveling calibration of 1/8delay till bit 7 that holdsthe result of the write-leveling calibration of 7/8 delay
23–16 HW_WL2_DQ	HW write-leveling calibration result of Byte2. This field holds the results for all the 8 write-leveling steps of Byte2. i.e bit 0 holds the result of the write-leveling calibration of 0 delay, bit 1holds the result of the write-leveling calibration of 1/8delay till bit 7 that holds the result of the write-leveling calibration of 7/8 delay
15–8 HW_WL1_DQ	HW write-leveling calibration result of Byte1. This field holds the results for all the 8 write-leveling steps of Byte1. i.e bit 0 holds the result of the write-leveling calibration of 0 delay, bit 1holds the result of the write-leveling calibration of 1/8delay till bit 7 that holds the result of the write-leveling calibration of 7/8 delay
HW_WL0_DQ	HW write-leveling calibration result of Byte0. This field holds the results for all the 8 write-leveling steps of Byte0. i.e bit 0 holds the result of the write-leveling calibration of 0 delay, bit 1holds the result of the write-leveling calibration of 1/8delay till bit 7 that holds the result of the write-leveling calibration of 7/8 delay

44.12.62 MMDC PHY Read DQS Gating HW Status Register 0 (MMDCx_MPDGHWST0)

Supported Mode Of Operations:

For Channel 0: DDR3_x16, DDR3_x32, DDR3_x64

For Channel 1: DDR3_x64

Address: Base address + 87Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved					HW_DG_UP0											Reserved					HW_DG_LOW0										
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDCx_MPDGHWST0 field descriptions

Field	Description
31–27 -	This field is reserved. Reserved
26–16 HW_DG_UP0	HW DQS gating calibration result of the upper boundary of Byte0. This field holds the HW DQS gating calibration result of the upper boundary of Byte0.
15–11 -	This field is reserved. Reserved
HW_DG_LOW0	HW DQS gating calibration result of the lower boundary of Byte0. This field holds the HW DQS gating calibration result of the lower boundary of Byte0.

44.12.63 MMDC PHY Read DQS Gating HW Status Register 1 (MMDCx_MPDGHWST1)

Supported Mode Of Operations:

For Channel 0: DDR3_x16, DDR3_x32, DDR3_x64

For Channel 1: DDR3_x64

Address: Base address + 880h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved					HW_DG_UP1											Reserved					HW_DG_LOW1										
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDCx_MPDGHWST1 field descriptions

Field	Description
31–27 -	This field is reserved. Reserved
26–16 HW_DG_UP1	HW DQS gating calibration result of the upper boundary of Byte1. This field holds the HW DQS gating calibration result of the upper boundary of Byte1.
15–11 -	This field is reserved. Reserved
HW_DG_LOW1	HW DQS gating calibration result of the lower boundary of Byte1. This field holds the HW DQS gating calibration result of the lower boundary of Byte1.

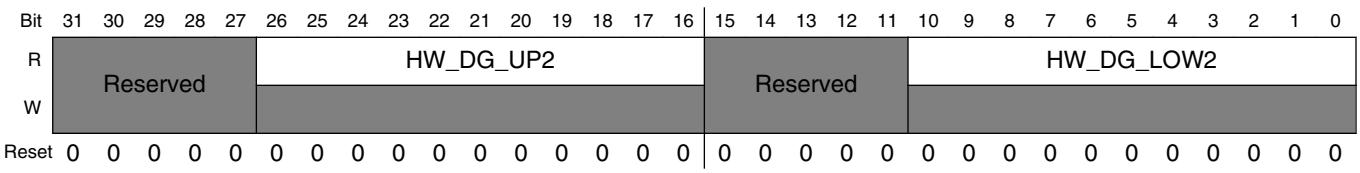
44.12.64 MMDC PHY Read DQS Gating HW Status Register 2 (MMDCx_MPDGHWST2)

Supported Mode Of Operations:

For Channel 0: DDR3_x16, DDR3_x32, DDR3_x64

For Channel 1: DDR3_x64

Address: Base address + 884h offset



MMDCx_MPDGHWST2 field descriptions

Field	Description
31–27 -	This field is reserved. Reserved
26–16 HW_DG_UP2	HW DQS gating calibration result of the upper boundary of Byte2. This field holds the HW DQS gating calibration result of the upper boundary of Byte2.
15–11 -	This field is reserved. Reserved
HW_DG_LOW2	HW DQS gating calibration result of the lower boundary of Byte2. This field holds the HW DQS gating calibration result of the lower boundary of Byte2.

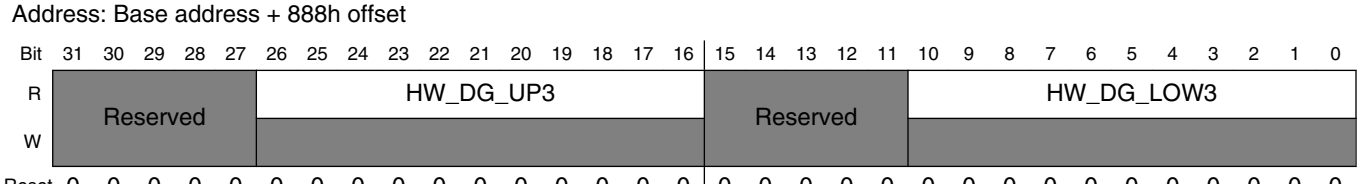
44.12.65 MMDC PHY Read DQS Gating HW Status Register 3 (MMDCx_MPDGHWST3)

Supported Mode Of Operations:

For Channel 0: DDR3_x16, DDR3_x32, DDR3_x64

For Channel 1: DDR3_x64

Address: Base address + 888h offset



MMDCx_MPDGHWST3 field descriptions

Field	Description
31–27 -	This field is reserved. Reserved
26–16 HW_DG_UP3	HW DQS gating calibration result of the upper boundary of Byte3. This field holds the HW DQS gating calibration result of the upper boundary of Byte3.
15–11 -	This field is reserved. Reserved
HW_DG_LOW3	HW DQS gating calibration result of the lower boundary of Byte3. This field holds the HW DQS gating calibration result of the lower boundary of Byte3.

44.12.66 MMDC PHY Pre-defined Compare Register 1 (MMDCx_MPPDCMPR1)

This register holds the MMDC pre-defined compare value that will be used during automatic read, read DQS gating and write calibration process. The compare value can be the MPR value (as defined in the JEDEC) or can be programmed by the PDV1 and PDV2 fields. In case of DDR3 (BL=8) the MMDC will duplicate PDV1,PDV2 and drive that data on Beat4-7 of the same byte

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 88Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PDV2																PDV1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDCx_MPPDCMPR1 field descriptions

Field	Description
31–16 PDV2	MMDC Pre defined compare value2. This field holds the 2 MSB of the data that will be driven to the DDR device during automatic read, read DQS gating and write calibrations in case MPR(DDR3)/ DQ calibration (LPDDR2) mode are disabled (MPR_CMP is disabled). Upon read access during the calibration the MMDC will compare the read data with the data that is stored in this field. Note : Before issue the read access the MMDC will invert the value of this field and drive it to the associate entry in the read comparison FIFO. For further information see Section 19.14.3.1.2, "Calibration with pre-defined value , Section 19.14.4.1.2, "Calibration with pre-defined value and Section 19.14.5.1, "HW (automatic) Write Calibraion
PDV1	MMDC Pre defined compare value2. This field holds the 2 LSB of the data that will be driven to the DDR device during automatic read, read DQS gating and write calibrations in case MPR(DDR3)/ DQ calibration (LPDDR2) mode are disabled (MPR_CMP is disabled). Upon read access during the calibration the MMDC will compare the read data with the data that is stored in this field.

Table continues on the next page...

MMDCx_MPPDCMPR1 field descriptions (continued)

Field	Description
	NOTE: Before issuing the read access, the MMDC will invert the value of this field and drive it to the associated entry in the read comparison FIFO.

44.12.67 MMDC PHY Pre-defined Compare and CA delay-line Configuration Register (MMDCx_MPPDCMPR2)

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 890h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	PHY_CA_DL_UNIT							0	CA_DL_ABS_OFFSET						
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0													READ_LEVEL_PATTERN	MPR_FULL_CMP	MPR_CMP
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDCx_MPPDCMPR2 field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30–24 PHY_CA_DL_UNIT	This field reflects the number of delay units that are actually used by CA (Command/Address of LPDDR2) delay-line
23 Reserved	This read-only field is reserved and always has the value 0.
22–16 CA_DL_ABS_OFFSET	Absolute CA (Command/Address of LPDDRR2) offset. This field indicates the absolute delay between CA (Command/Address) bus and the DDR clock (CK) with fractions of a clock period and up to half cycle. The fraction is process and frequency independent. The delay of the delay-line would be (CA_DL_ABS_OFFSET / 256) * MMDC_CH0 AXI clock (fast clock). So for the default value of 64 we get a quarter cycle delay.
15–3 Reserved	This read-only field is reserved and always has the value 0.
2 READ_LEVEL_PATTERN	MPR(DDR3)/DQ calibration(LPDDR2) read compare pattern. In case MPR(DDR3)/DQ calibration(LPDDR2) modes are used during the calibration process (MPR_CMP is asserted) then this field indicates the read pattern for the comparison. 0 Compare with read pattern 1010 1 Compare with read pattern 0011 (Used only in LPDDR2 mode)
1 MPR_FULL_CMP	MPR(DDR3)/DQ calibration (LPDDR2) full compare enable. In case MPR(DDR3)/DQ calibration(LPDDR2) modes are used during the calibration process (MPR_CMP is asserted) then this field indicates whether the MMDC will compare all the bits of the data that is read from the DDR device to the MPR pre-defined pattern. When this bit is de-asserted only LSB of each byte is compared.
0 MPR_CMP	MPR(DDR3)/DQ calibration (LPDDR2) compare enable. This bit indicates whether the MMDC will compare the read data during automatic read and read DQS calibration processes to the pre-defined patterns that are driven by the DDR deivce (READ_LEVEL_PATTERN as defined by JEDEC) or general pre-defined value that are stored in PDV1 and PDV2. When this bit is disabled data is compared to the data of the pre defined compare value field For further information see Read DQS Gating Calibration and Read Calibration .

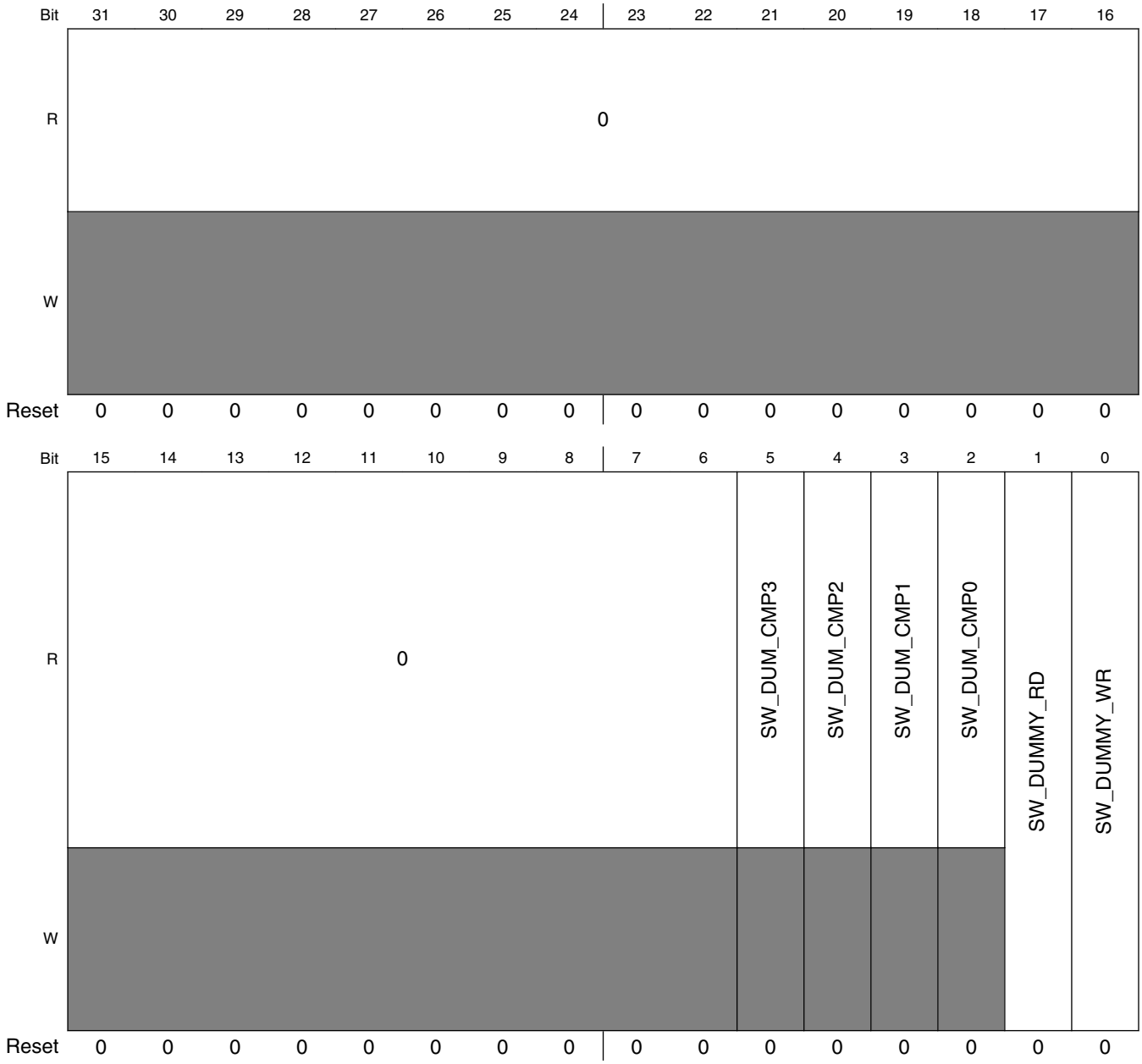
44.12.68 MMDC PHY SW Dummy Access Register (MMDCx_MPSWDAR0)

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: DDR3_x64, LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 894h offset



MMDCx_MPSWDAR0 field descriptions

Field	Description
31–6 Reserved	This read-only field is reserved and always has the value 0.
5 SW_DUM_CMP3	SW dummy read byte3 compare results. This bit indicates the result of the read data comparison of Byte3 at the completion of SW_DUMMY_RD. This bit is valid only when SW_DUMMY_RD is de-assrted. 0 Dummy read fail 1 Dummy read pass
4 SW_DUM_CMP2	SW dummy read byte2 compare results. This bit indicates the result of the read data comparison of Byte2 at the completion of SW_DUMMY_RD. This bit is valid only when SW_DUMMY_RD is de-assrted.

Table continues on the next page...

MMDCx_MPSWDAR0 field descriptions (continued)

Field	Description
	0 Dummy read fail 1 Dummy read pass
3 SW_DUM_CMP1	SW dummy read byte1 compare results. This bit indicates the result of the read data comparison of Byte1 at the completion of SW_DUMMY_RD. This bit is valid only when SW_DUMMY_RD is de-assrted. 0 Dummy read fail 1 Dummy read pass
2 SW_DUM_CMP0	SW dummy read byte0 compare results. This bit indicates the result of the read data comparison of Byte0 at the completion of SW_DUMMY_RD. This bit is valid only when SW_DUMMY_RD is de-assrted. 0 Dummy read fail 1 Dummy read pass
1 SW_DUMMY_ RD	SW dummy read. When this bit is asserted the MMDC will generate internally read access without intervention of the system toward bank 0, row 0, column 0. If MPR_CMP = 1then the read data will be compared to MPPDCMPR2[READ_LEVEL_PATTERN] . If MPR_CMP =0 then the read data will be compared to MPPDCMPR1[PDV1], MPPDCMPR1[PDV2]. Upon completion of the access this bit is de-asserted automatically and the read data and comparison results are valid at MPSWDAR0[SW_DUM_CMP#] and MPSWDRDR0-MPSWDRDR7 respectively.
0 SW_DUMMY_ WR	SW dummy write. When this bit is asserted the MMDC will generate internally write access without intervention of the system toward bank 0, row 0, column 0, while the data is driven from MPPDCMPR1[PDV1] and MPPDCMPR1[PDV2]. The bit is de-asserted automatically upon completion of the access.

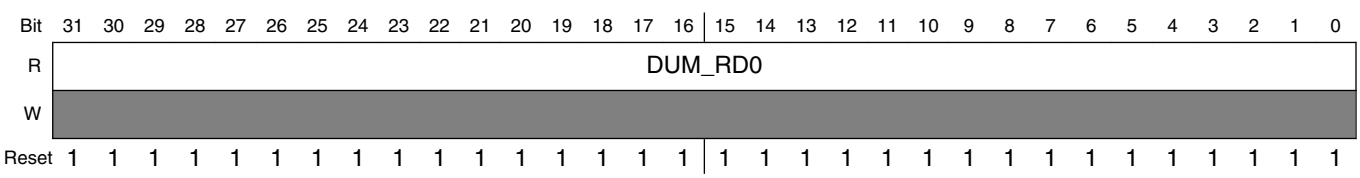
44.12.69 MMDC PHY SW Dummy Read Data Register 0 (MMDCx_MPSWDRDR0)

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: DDR3_x64, LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 898h offset



MMDCx_MPSWDRDR0 field descriptions

Field	Description
DUM_RD0	Dummy read data0. This field holds the first data that is read from the DDR during SW dummy read access (i.e when SW_DUMMY_RD = 1). This field is valid only when SW_DUMMY_RD is de-assrted

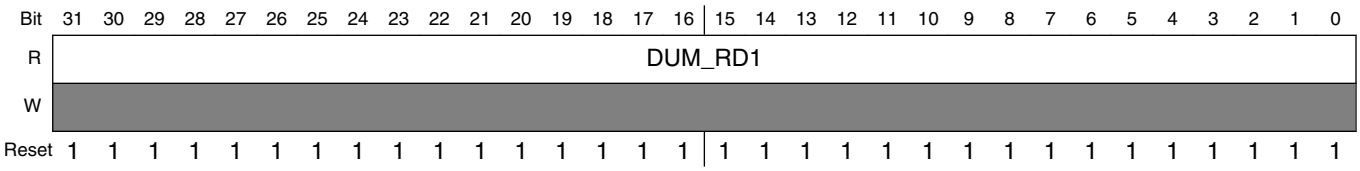
44.12.70 MMDC PHY SW Dummy Read Data Register 1 (MMDCx_MPSWDRDR1)

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: DDR3_x64, LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 89Ch offset



MMDCx_MPSWDRDR1 field descriptions

Field	Description
DUM_RD1	Dummy read data1. This field holds the second data that is read from the DDR during SW dummy read access (i.e when SW_DUMMY_RD = 1). This field is valid only when SW_DUMMY_RD is de-assrted

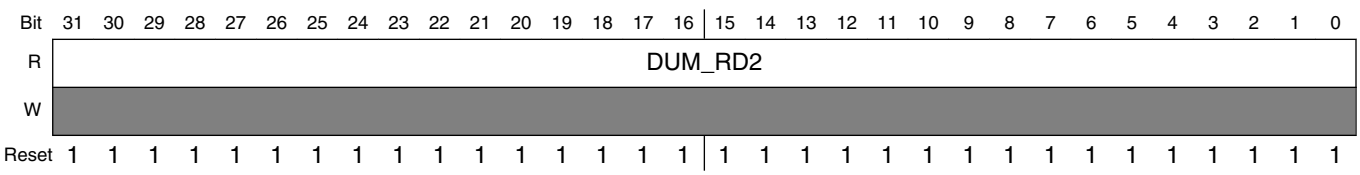
44.12.71 MMDC PHY SW Dummy Read Data Register 2 (MMDCx_MPSWDRDR2)

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: DDR3_x64, LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 8A0h offset



MMDCx_MPSWDRDR2 field descriptions

Field	Description
DUM_RD2	Dummy read data2. This field holds the third data that is read from the DDR during SW dummy read access (i.e when SW_DUMMY_RD = 1). This field is valid only when SW_DUMMY_RD is de-assrted.

44.12.72 MMDC PHY SW Dummy Read Data Register 3 (MMDCx_MPSWDRDR3)

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: DDR3_x64, LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 8A4h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DUM_RD3																															
W																																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

MMDCx_MPSWDRDR3 field descriptions

Field	Description
DUM_RD3	Dummy read data3. This field holds the forth data that is read from the DDR during SW dummy read access (i.e when SW_DUMMY_RD = 1). This field is valid only when SW_DUMMY_RD is de-assrted.

44.12.73 MMDC PHY SW Dummy Read Data Register 4 (MMDCx_MPSWDRDR4)

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: DDR3_x64, LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 8A8h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DUM_RD4																															
W																																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

MMDCx_MPSWDRDR4 field descriptions

Field	Description
DUM_RD4	Dummy read data4. This field holds the fifth data (only in case of burst length 8 (BL =1)) that is read from the DDR during SW dummy read access (i.e when SW_DUMMY_RD = 1). This field is valid only when SW_DUMMY_RD is de-assrted.

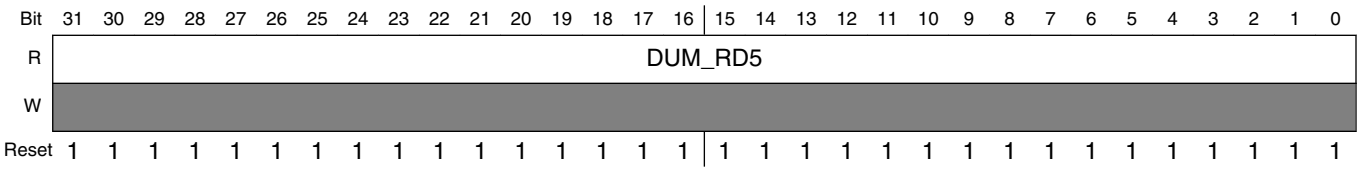
44.12.74 MMDC PHY SW Dummy Read Data Register 5 (MMDCx_MPSWDRDR5)

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: DDR3_x64, LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 8ACh offset



MMDCx_MPSWDRDR5 field descriptions

Field	Description
DUM_RD5	Dummy read data5. This field holds the sixth data (only in case of burst length 8 (BL =1)) that is read from the DDR during SW dummy read access (i.e when SW_DUMMY_RD = 1). This field is valid only when SW_DUMMY_RD is de-assrted.

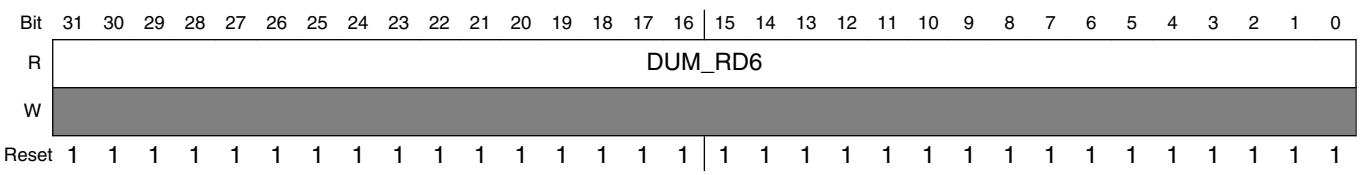
44.12.75 MMDC PHY SW Dummy Read Data Register 6 (MMDCx_MPSWDRDR6)

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: DDR3_x64, LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 8B0h offset



MMDCx_MPSWDRDR6 field descriptions

Field	Description
DUM_RD6	Dummy read data6. This field holds the seventh data (only in case of burst length 8 (BL =1)) that is read from the DDR during SW dummy read access (i.e when SW_DUMMY_RD = 1). This field is valid only when SW_DUMMY_RD is de-assrted.

44.12.76 MMDC PHY SW Dummy Read Data Register 7 (MMDCx_MPSWDRDR7)

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: DDR3_x64, LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 8B4h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DUM_RD7																															
W																																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

MMDCx_MPSWDRDR7 field descriptions

Field	Description
DUM_RD7	Dummy read data7. This field holds the eigth data (only in case of burst length 8 (BL =1)) that is read from the DDR during SW dummy read access (i.e when SW_DUMMY_RD = 1). This field is valid only when SW_DUMMY_RD is de-assrted.

44.12.77 MMDC PHY Measure Unit Register (MMDCx_MPMUR0)

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 8B8h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0						MU_UNIT_DEL_NUM									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				FRC_MSR	MU_BYP_EN	MU_BYP_VAL									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDCx_MPMUR0 field descriptions

Field	Description
31–26 Reserved	This read-only field is reserved and always has the value 0.
25–16 MU_UNIT_DEL_NUM	Number of delay units measured per cycle. This field is used in debug mode and holds the number of delay units that were measured by the measure unit per DDR clock cycle. The delay-lines that are used in every calibration process use that number for generating the desired delay.
15–12 Reserved	This read-only field is reserved and always has the value 0.
11 FRC_MSR	Force measuement on delay-lines. When this bit is asserted then a measurement process will be performed, where at the completion of the process the delay-lines will issue the desired delay. Upon completion of the measurement process the measure unit and the delay-lines will return to functional mode. This bit is self cleared. **NOTE:** This bit should be used only during manual (SW) calibration and not while the DDR is functional (being accessed). After initial calibration is done the hardware performs periodic measurements to track any operating conditions changes. Hence, force measurements (FRC_MSR) should not be used. See Calibration Process for more information. **NOTE:** User should make sure that there is no active accesses to/from DDR before asserting this bit. 0 No measurement is performed 1 Perform measurement process
10 MU_BYP_EN	Measure unit bypass enable. This field is used in debug mode and when it is asserted then the delay-lines will use the number of delay units that are indicated at MU_BYP_VAL, otherwise the delay-lines will use the number of delay units that was measured by the measurement unit and are indicated at MU_UNIT_DEL_NUM 0 The delay-lines use delay units as indicated at MU_UNIT_DEL_NUM. 1 The delay-lines use delay units as indicated at MU_BYPASS_VAL.
MU_BYP_VAL	Number of delay units for measurement bypass. This field is used in debug mode and holds the number of delay units that will be used by the delay-lines when MU_BYP_EN is asserted.

44.12.78 MMDC Write CA delay-line controller (MMDCx_MPWRCADL)

This register is used to add fine-tuning adjustment to the CA (command/Address of LPDDR2 bus) relative to the DDR clock

Supported Mode Of Operations:

For Channel 0: LP2_2ch_x16, LP2_2ch_x32

For Channel 1: LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 8BCh offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0												WR_CA9_DEL		WR_CA8_DEL	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WR_CA7_DEL		WR_CA6_DEL		WR_CA5_DEL		WR_CA4_DEL		WR_CA3_DEL		WR_CA2_DEL		WR_CA1_DEL		WR_CA0_DEL	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MMDCx_MPWRCADL field descriptions

Field	Description
31–20 Reserved	This read-only field is reserved and always has the value 0.
19–18 WR_CA9_DEL	CA (Command/Address LPDDR2 bus) bit 9 delay fine tuning. This field holds the number of delay units that are added to CA (Command/Address bus) bit 9 relative to the clock. 00 No change in CA9 delay 01 Add CA9 delay of 1 delay unit 10 Add CA9 delay of 2 delay units. 11 Add CA9 delay of 3 delay units.
17–16 WR_CA8_DEL	CA (Command/Address LPDDR2 bus) bit 8 delay fine tuning. This field holds the number of delay units that are added to CA (Command/Address bus) bit 8 relative to the clock. 00 No change in CA8 delay 01 Add CA8 delay of 1 delay unit 10 Add CA8 delay of 2 delay units. 11 Add CA8 delay of 3 delay units.
15–14 WR_CA7_DEL	CA (Command/Address LPDDR2 bus) bit 7 delay fine tuning. This field holds the number of delay units that are added to CA (Command/Address bus) bit 7 relative to the clock. 00 No change in CA7 delay 01 Add CA7 delay of 1 delay unit 10 Add CA7 delay of 2 delay units. 11 Add CA7 delay of 3 delay units.
13–12 WR_CA6_DEL	CA (Command/Address LPDDR2 bus) bit 6 delay fine tuning. This field holds the number of delay units that are added to CA (Command/Address bus) bit 6 relative to the clock. 00 No change in CA6 delay 01 Add CA6 delay of 1 delay unit 10 Add CA6 delay of 2 delay units. 11 Add CA6 delay of 3 delay units.
11–10 WR_CA5_DEL	CA (Command/Address LPDDR2 bus) bit 5 delay fine tuning. This field holds the number of delay units that are added to CA (Command/Address bus) bit 5 relative to the clock. 00 No change in CA5 delay 01 Add CA5 delay of 1 delay unit 10 Add CA5 delay of 2 delay units. 11 Add CA5 delay of 3 delay units.
9–8 WR_CA4_DEL	CA (Command/Address LPDDR2 bus) bit 4 delay fine tuning. This field holds the number of delay units that are added to CA (Command/Address bus) bit 4 relative to the clock. 00 No change in CA4 delay 01 Add CA4 delay of 1 delay unit 10 Add CA4 delay of 2 delay units. 11 Add CA4 delay of 3 delay units.

Table continues on the next page...

MMDCx_MPWRCADL field descriptions (continued)

Field	Description
7–6 WR_CA3_DEL	CA (Command/Address LPDDR2 bus) bit 3 delay fine tuning. This field holds the number of delay units that are added to CA (Command/Address bus) bit 3 relative to the clock. 00 No change in CA3 delay 01 Add CA3 delay of 1 delay unit 10 Add CA3 delay of 2 delay units. 11 Add CA3 delay of 3 delay units.
5–4 WR_CA2_DEL	CA (Command/Address LPDDR2 bus) bit 2 delay fine tuning. This field holds the number of delay units that are added to CA (Command/Address bus) bit 2 relative to the clock. 00 No change in CA2 delay 01 Add CA2 delay of 1 delay unit 10 Add CA2 delay of 2 delay units. 11 Add CA2 delay of 3 delay units.
3–2 WR_CA1_DEL	CA (Command/Address LPDDR2 bus) bit 1 delay fine tuning. This field holds the number of delay units that are added to CA (Command/Address bus) bit 1 relative to the clock. 00 No change in CA1 delay 01 Add CA1 delay of 1 delay unit 10 Add CA1 delay of 2 delay units. 11 Add CA1 delay of 3 delay units.
WR_CA0_DEL	CA (Command/Address LPDDR2 bus) bit 0 delay fine tuning. This field holds the number of delay units that are added to CA (Command/Address bus) bit 0 relative to the clock. 00 No change in CA0 delay 01 Add CA0 delay of 1 delay unit 10 Add CA0 delay of 2 delay units. 11 Add CA0 delay of 3 delay units.

44.12.79 MMDC Duty Cycle Control Register (MMDCx_MPDCCR)

This register is used to control the duty cycle of the DQS and the primary clock (CK0) . Programming of that register is permitted by entering the DDR device into self-refresh mode through LPMD/DVFS mechanism

Supported Mode Of Operations:

For Channel 0: All

For Channel 1: DDR3_x64, LP2_2ch_x16, LP2_2ch_x32

Address: Base address + 8C0h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	RD_DQS3_FT_DCC			RD_DQS2_FT_DCC			RD_DQS1_FT_DCC			RD_DQS0_FT_DCC			CK_FT1_DCC		
W																
Reset	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	CK_FT0_DCC			WR_DQS3_FT_DCC			WR_DQS2_FT_DCC			WR_DQS1_FT_DCC			WR_DQS0_FT_DCC		
W																
Reset	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0

MMDCx_MPDCCR field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30–28 RD_DQS3_FT_DCC	Read DQS duty cycle fine tuning control of Byte3. This field controls the duty cycle of read DQS of Byte3 Note all the other options are not allowed 001 48.5% low 51.5% high 010 50% duty cycle (default) 100 51.5% low 48.5% high
27–25 RD_DQS2_FT_DCC	Read DQS duty cycle fine tuning control of Byte2. This field controls the duty cycle of read DQS of Byte2 Note all the other options are not allowed 001 48.5% low 51.5% high 010 50% duty cycle (default) 100 51.5% low 48.5% high
24–22 RD_DQS1_FT_DCC	Read DQS duty cycle fine tuning control of Byte1. This field controls the duty cycle of read DQS of Byte1 Note all the other options are not allowed 001 48.5% low 51.5% high 010 50% duty cycle (default) 100 51.5% low 48.5% high
21–19 RD_DQS0_FT_DCC	Read DQS duty cycle fine tuning control of Byte0. This field controls the duty cycle of read DQS of Byte0 Note all the other options are not allowed 001 48.5% low 51.5% high 010 50% duty cycle (default) 100 51.5% low 48.5% high
18–16 CK_FT1_DCC	Secondary duty cycle fine tuning control of DDR clock. This field controls the duty cycle of the DDR clock and is cascaded to CK_FT0_DCC Note all the other options are not allowed 001 48.5% low 51.5% high 010 50% duty cycle (default) 100 51.5% low 48.5% high
15 Reserved	This read-only field is reserved and always has the value 0.
14–12 CK_FT0_DCC	Primary duty cycle fine tuning control of DDR clock. This field controls the duty cycle of the DDR clock Note all the other options are not allowed 001 48.5% low 51.5% high

Table continues on the next page...

MMDCx_MPDCCR field descriptions (continued)

Field	Description
	010 50% duty cycle (default) 100 51.5% low 48.5% high
11–9 WR_DQS3_FT_DCC	Write DQS duty cycle fine tuning control of Byte0. This field controls the duty cycle of write DQS of Byte0 Note all the other options are not allowed 001 48.5% low 51.5% high 010 50% duty cycle (default) 100 51.5% low 48.5% high
8–6 WR_DQS2_FT_DCC	Write DQS duty cycle fine tuning control of Byte1. This field controls the duty cycle of write DQS of Byte1 Note all the other options are not allowed 001 48.5% low 51.5% high 010 50% duty cycle (default) 100 51.5% low 48.5% high
5–3 WR_DQS1_FT_DCC	Write DQS duty cycle fine tuning control of Byte1. This field controls the duty cycle of write DQS of Byte1 Note all the other options are not allowed 001 48.5% low 51.5% high 010 50% duty cycle (default) 100 51.5% low 48.5% high
WR_DQS0_FT_DCC	Write DQS duty cycle fine tuning control of Byte0. This field controls the duty cycle of write DQS of Byte0 Note all the other options are not allowed 001 48.5% low 51.5% high 010 50% duty cycle (default) 100 51.5% low 48.5% high

Chapter 45
Network Interconnect Bus System (NIC-301)

45.1 Overview

This section provides an overview of the NIC-301 (Network Inter-Connect) AXI arbiter IP.

The NIC-301 (by ARM Ltd.) is a configurable AXI arbiter between several masters and slaves. The NIC-301 IP is designed so that many configuration options are selected at the hardware design stage, determined by SoC characteristics and needs, while several other configuration options are software-controlled.

This chapter covers in brief the NIC-301 functionality, while providing configuration details on the NIC-301 instances used in the chip. For complete details on the NIC-301 design, see the ARM specification, *AMBA® Network Interconnect (NIC-301) Technical Reference Manual*.

NOTE

The NIC-301 default settings are configured by Freescale's board support package (BSP), and in most cases should not be modified by the customer. The default settings have gone through exhaustive testing during the validation of the part, and have proven to work well for the part's intended target applications. Changes to the default settings may result in a degradation in system performance.

45.2 Block diagram

The NIC-301 AXI arbiter (or "CoreLink Network Interconnect") by ARM, provides configurable AXI-based interconnect logic, for connecting a number of masters (initiators) to several slaves (targets), via a configurable bus switches and bridging components.

The bus system is composed of five such instances: PER1, PER2, FAST1, FAST2, and FAST3.

Each instance can include one or more bus switches, with additional logic.

This chapter provides details of the various instances and the selected configuration parameters.

The top level diagram of the bus system is shown in the following figure.

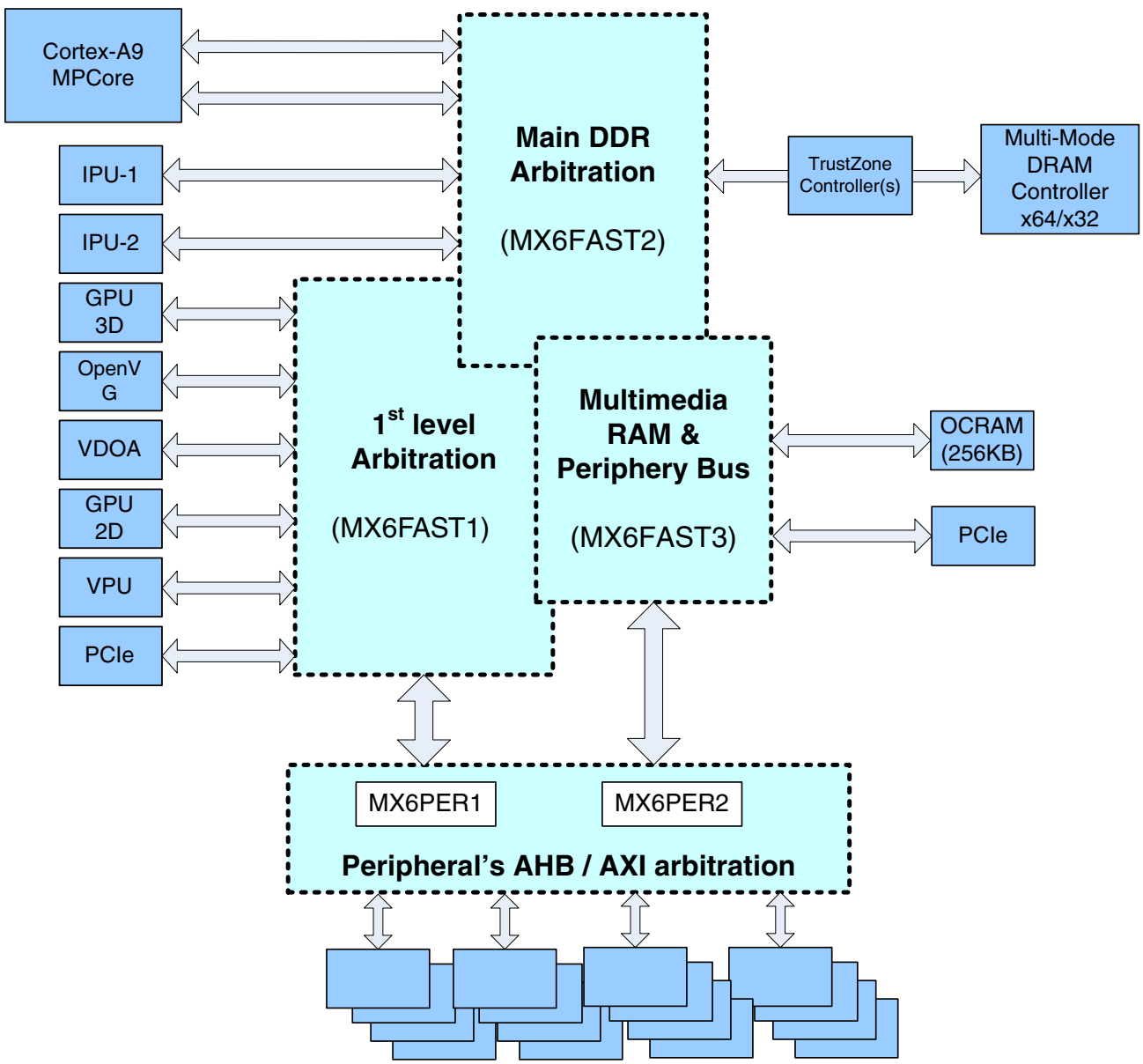


Figure 45-1. NIC-301 Bus System

45.2.1 NIC-301 Main Features

Key features of the NIC-301 module include the following:

- Address space memory mapping, including 'remap' functions.
- Programmer's view, for software-configured parameters, via "GPV" ports.
- Support for cross-clock domain synchronization.

45.2.2 Modes and Operations

The NIC-301 supports a normal functional mode only, as described in Normal Mode.

45.3 External Signals

The NIC-301 has no external I/O interfaces.

45.4 Memory Map and Register Definition

This section includes the block memory map and detailed register descriptions.

Access to NIC-301 registers is provided through the global programmer's view (GPV) ports. Each GPV port provides access to the configuration registers of certain IP as listed in Table 45-5 . The GPV base addresses are listed in the table below:

Table 45-1. GPV ports memory allocations

GPV	Associated NIC-301	System Bus	Chip Address	
			Start	End
GPV_0	FAST2	Main DDR Arbitration	00B0_0000	00BF_FFFF
GPV_1	FAST1	First Level Arbitration	00C0_0000	00CF_FFFF
GPV_2	PER1	Peripheral's AHB/AXI Arbitration	0020_0000	002F_FFFF
GPV_3	PER2	Reserved (Internal Use Only)	0030_0000	003F_FFFF
GPV_4	FAST3	Multimedia RAM and Peripheral Bus	0080_0000	008F_FFFF

45.4.1 Memory Map

The NIC-301 memory map, is dependent on the selected configuration option at time of creation.

A "template" map is provided in Table 45-2 below. For specific features, see the configuration tables in NIC-specific parameters to check whether specific options are selected.

45.4.2 Configuration programmers model

The GPV's contain configuration registers, partitioned into a number of individual 4KB blocks.

The general structure of the registers is provided by the following tables:

- Address map of the programmers model, Table 45-2
- AMIB Registers, Table 45-3
- ASIB Registers, Table 45-4

Table 45-2. Address map of the programmers model

Address Offset from Base Address	Registers	Notes
0x000F_F000	Internal interface p registers	Maximum p = 61
	...	Note [1]
0x000C_4000	Internal interface 2 registers	
0x000C_3000	Internal interface 1 registers	
0x000C_2000	Internal interface 0 registers	
0x000C_1000	Slave interface m registers	Maximum m = 127
	...	Note [2]
0x0004_4000	Slave interface 2 registers	
0x0004_3000	Slave interface 1 registers	
0x0004_2000	Slave interface 0 registers	
0x0004_1000	Master interface n registers	Maximum n = 63
	...	Note [3]
0x0000_4000	Master interface 2 registers	
0x0000_3000	Master interface 1 registers	
0x0000_2000	Master interface 0 registers	
0x0000_1000	ID registers	
0x0000_0000	Address control registers	Configurable base address [4]

1. Index refers to BI registers index
2. Index refer to ASIB registers index
3. Index refer to AMIB registers index

4. Reserved for internal use

45.4.2.1 Address control and ID registers

Registers at offsets 0x0–0xFFC are reserved for internal use.

45.4.2.2 AMBA master interface block (AMIB) configuration registers

The table below lists only the registers that affect the user. All other addresses are treated as "reserved".

Table 45-3. AMIB Registers

Offset	Register	Access	Width	Reset Value
0x024	fn_mod2 Bypass merge. This register is only present if upsizing or downsizing. See upsizing/downsizing data width functions in AMBA Network Interconnect TRM.	RW	1	0
0x040	wr_tidemark	RW	4	Note [1]

1. Reset value varies, default value chosen at RTL creation time is designed to suit normal operation.

45.4.2.3 ASIB (AMBA slave interface block) configuration registers

The table below lists only the registers that affect the user. All other addresses are treated as "reserved".

Table 45-4. ASIB Registers

Offset	Register	Access	Width	Reset Value
0x040	wr_tidemark Valid only for AXI slaves with WFIFO >=4.	RW	4	Note [1]
0x100	read_qos	RW	4	Note [2]
0x104	write_qos	RW	4	Note4

1. Reset value varies, default value chosen at RTL creation time is designed to suit typical operation cases.
2. QoS default is set at RTL creation time, and is listed in specific NIC-301 configuration tables in NIC-specific parameters, as parameters "QoS qv_value" in ASIB / AMIB parameter tables.

45.4.3 Register Descriptions

The NIC-301 registers are dependent upon the selected configuration, the type of ports, hardware-selected features and whether they have a GPV view. The addressing is associated to a specific port, by looking at the port's index number, under "apb_slave" column.

The memory map template is provided in the Configuration programmers model above.

45.4.3.1 QoS registers' address look-up example

The flow below describes the steps needed in determining the memory addresses for GPU3D QoS read/write registers:

- Look up the GPU3D port indexes ("apb_slave"[1] column in Table 45-5): index 66 and 71 for primary and secondary buses, respectively. Note that the GPU3D is listed under GPV_1 interface.
- Look up the GPV_1 base address: 0x00C0_0000, in Table 45-1 .
- The master/slave ports configuration base address, is always obtained by: "GPV base" + ("APB index" x 0x1000), regardless of whether it is a slave or a master port. This is somewhat contrary to the way indexes are presented in the "Address map of the NIC-301 programmers model" in the ARM documentation as shown in Table 45-2, where, instead, the indexes are provided in absolute values (of 0x1000 jumps), starting from the GPV base address.

  Hence, for the GPU3D primary port (index '66'), the "slave interface" programming base address is 0x00C4_2000 (= GPV_1 base + 66 x 0x1000).

  Similarly, for the GPU3D's secondary port (index 71), the "slave interface" programming base address is 0x00C4_7000 (= GPV_1 base + 71 x 0x1000).

- To access any of the slave/master interface registers (in this example - the QoS read/write ones), their offset(s) must be added, as shown in Table 45-4 :

  "0x100" for Read_QoS and "0x104" for Write_QoS, in our example.

Final addresses obtained for QoS registers for the GPU3D primary port (index 66), are as follows:

- Read QoS: "0x00C4_2100"
- Write QoS: "0x00C4_2104"

1. The "slave", "master" type is assumed from NIC-301 point of view.

Similarly, QoS registers addresses for the GPU3D secondary port (index 71), are as follows:

- Read QoS: "0x00C4_7100"
- Write QoS: "0x00C4_7104"

45.4.4 NIC-specific parameters

This section details the configuration parameters of the NIC-301.

General notes:

1. All accesses to GPV_4 port, must be of "Supervisor" type.
2. The associated master/slave port interface ID (for each "GPV_N") is specified by the "apb_slave" / "apb_master" [2] index. The slave/master interface, configuration register's start address is obtained by the following:

   Start address (for "apb" index= "n") = GPV_N base + (n x 0x1000).

3. The security features of NIC-301 are not used, since master-slaves permissions are controlled by the CSU security policy/scheme.

Table 45-5. QoS and tidemark parameters

Master	apb_slave	Tidemark	QOS qv_value	Comments
Configured via GPV_1 port				
GPU3D (Primary)	66	2	2	
GPU3D (Secondary)	71	2	2	
GPU2D (Primary)	67	2	2	
GPU2D (Secondary)	72	2	2	
VDOA	68	2	2	
OpenVG	69	2	2	
VPU Prim.	73	2	2	
PCIe	74	2	2	
Configured via GPV_0 port				
MPCore-0	66		2	
MPCore-1	67		2	
IPU1	68	4	0	QoS configurable via GPR bits in IOMUXC.
IPU2	69	4	0	QoS configurable via GPR bits in IOMUXC.
Configured via GPV_4 port				
VPU	68		2	

Table continues on the next page...

2. "APB" stands for "ARM Peripheral Bus".

Table 45-5. QoS and tidemark parameters (continued)

Master	apb_slave	Tidemark	QOS qv_value	Comments
(to internal RAM)				
Configured via GPV_2 port (switch 0)				
HDMI	66		2	
SDMA burst	67		3	
SDMA Peripheral	68		3	
CAAM	69		2	
USBOH3A	70		2	
ENET	71		2	
HSI	72		2	
uSDHC1	73		2	
Configured via GPV_2 port (switch 1)				
DAP	74		2	
APBH	75		2	
BCH40	76		2	
SATA	77		2	
MLB150	78		2	
uSDHC2	79		2	
uSDHC3	80		2	
uSDHC4	81		2	

Chapter 46
On-Chip OTP Controller (OCOTP_CTRL)

46.1 Overview

This section contains information describing the requirements for the on-chip eFuse OTP controller along with details about the block functionality and implementation.

In this document, the words "eFuse" and "OTP" are interchangeable. OCOTP refers to the hardware block itself.

46.1.1 Features

The OCOTP provides the following features :

- 32-bit word restricted program and read to 4Kbits of eFuse OTP(512x8).
- Loading and housing of fuse content into shadow registers.
- memory-mapped (restricted) access to 4Kbits of shadow registers.
- Generation of hwv_fuse (hardware visible fuse bus) and the hwv_reg bus which is made of up of volatile PIO register based "fuses". The hwv_reg bits come from the SCS (Software Controllable Signals) register.
- Generation of sticky_reg which is consist of sticky register bits.
- Provide program-protect and read-protect efuse.
- Provide overide and read protection of shadow register.
- CRC32 test for read-lock fuse content.

46.2 Clocks

The table found here describes the clock sources for OCOTP.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 46-1. OCOTP Clocks

Clock name	Clock Root	Description
ipg_clk	ipg_clk_root	Peripheral clock
ipg_clk_s	ipg_clk_root	Peripheral access clock

46.3 Top-Level Symbol and Functional Overview

The figure found here shows the OCOTP system level diagram.

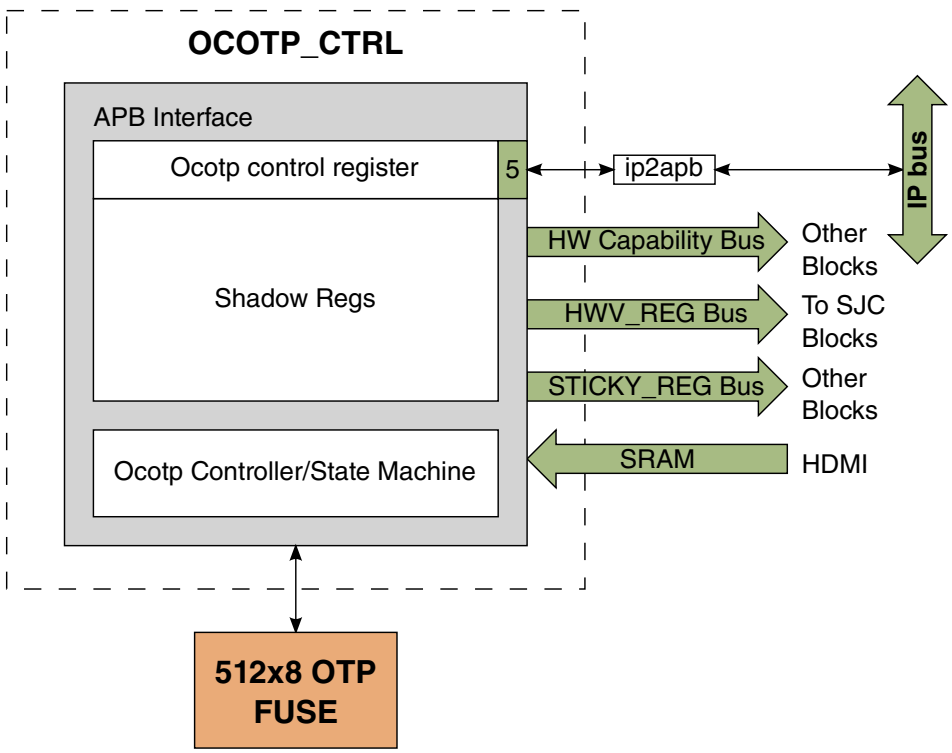


Figure 46-1. OCOTP System Level Diagram

46.3.1 Operation

The IP bus interface of the OCOTP provides two functions.

- Configure control registers for programming and reading fuse word.
- Overide and read shadow registers.

For efuse, program can only be performed on bit and read is based on byte. OCOTP configuration for program and read are performed on 32-bit words for SW convenience. For writes, the 32-bit word reflects the "write-mask". Bit fields with 0 will not be programmed and bit fields with 1 will be programmed. OCOTP will program bit field with 1 in the fuse word one bit by one bit. For reads, OCOTP will read 4 times to get 4 bytes in the fuse word in order.

In this document, 4k bits fuse are divided into 16 banks by function.Each bank has 8 fuse words.In physical, 4k bits fuse are in one 512x8 efusebox.

46.3.1.1 Shadow Register Reload

All fuse words in efusebox are shadowed. Therefore, fuse information is available through memory mapped shadow registers. If fuses are subsequently programmed, the shadow registers should be reloaded to keep them coherent with the fuse bank arrays.

The "reload shadows" feature allows the user to force a reload of the shadow registers (including HW_OCOTP_LOCK) without having to reset the device. To force a reload, complete the following steps:

1. Set the HW_OCOTP_TIMING[STROBE_READ] and HW_OCOTP_TIMING[RELAX] field value appropriately (as explained in a later section).
2. Check that HW_OCOTP_CTRL[BUSY] and HW_OCOTP_CTRL[ERROR] are clear. Overlapped accesses are not supported by the controller. Any pending write , read or reload must be completed before a new access can be requested.
3. Set the HW_OCOTP_CTRL[RELOAD_SHADOWS] bit. OCOTP will read all the fuse one by one and put it into corresponding shadow register.
4. Wait for HW_OCOTP_CTRL[BUSY] and HW_OCOTP_CTRL[RELOAD_SHADOWS] to be cleared by the controller.

The controller will automatically clear the HW_OCOTP_CTRL[RELOAD_SHADOWS] bit after the successful completion of the operation.

46.3.1.2 Fuse and Shadow register read

All shadow registers are always readable through the APB bus except some secret keys regions. When their corresponding fuse lock bits are set, the shadow registers also become read locked. After read locking, reading from these registers will return 0xBADABADA.

In addition HW_OCOTP_CTRL[ERROR] will be set. It must be cleared by software before any new write , read or reload access can be issued. Subsequent reads to unlocked shadow locations will still work successfully however.

To read fuse word directly from fusebox correctly complete the following steps:

1. Program HW_OCOTP_TIMING[STROBE_READ] and HW_OCOTP_TIMING[RELAX] fields with timing values to match the current frequency of the ipg_clk. OTP read will work at maximum bus frequencies as long as the HW_OCOTP_TIMING parameters are set correctly.
2. Check that HW_OCOTP_CTRL[BUSY] and HW_OCOTP_CTRL[ERROR] are clear. Overlapped accesses are not supported by the controller. Any pending write, read or reload must be completed before a read access can be requested.
3. Write the requested address to HW_OCOTP_CTRL[ADDR].
4. Set HW_OCOTP_READ_CTRL[READ_FUSE] to 1. OCOTP will auto read 4 bytes in requested word address in fusebox one by one. Then put read value into HW_OCOTP_READ_FUSE_DATA register.
5. Once complete, the controller will clear BUSY. A read request to a protected or locked region will result in no OTP access and no setting of HW_OCOTP_CTRL[BUSY]. In addition HW_OCOTP_CTRL[ERROR] will be set. It must be cleared by software before any new access can be issued.
6. Read HW_OCOTP_READ_FUSE_DATA register to get fuse word value.HW_OCOTP_READ_FUSE_DATA will be 0xBADABADA when HW_OCOTP_CTRL[ERROR] is set.

46.3.1.3 Fuse and Shadow Register Writes

Shadow register bits can be overridden by software until the corresponding fuse lock bit for the region is set. When the lock shadow bit is set, the shadow registers for that lock region become write locked. The LOCK shadow register also has no shadow or fuse lock bits but it is always read only.

In order to avoid "rogue" code performing erroneous writes to OTP, a special unlocking sequence is required for writes to the fuse banks. To program fuse bank correctly complete the following steps:

1. Program HW_OCOTP_TIMING[STROBE_PROG] and HW_OCOTP_TIMING[RELAX] fields with timing values to match the current frequency of the ipg_clk. OTP writes will work at maximum bus frequencies as long as the HW_OCOTP_TIMING parameters are set correctly.

2. Check that HW_OCOTP_CTRL[BUSY] and HW_OCOTP_CTRL[ERROR] are clear. Overlapped accesses are not supported by the controller. Any pending write or reload must be completed before a write access can be requested.
3. Write the requested address to HW_OCOTP_CTRL[ADDR] and program the unlock code into HW_OCOTP_CTRL[WR_UNLOCK]. This must be programmed for each write access. The lock code is documented in the register description. Both the unlock code and address can be written in the same operation.
4. Write the data to the HW_OCOTP_DATA register. This will automatically set HW_OCOTP_CTRL[BUSY] and clear HW_OCOTP_CTRL[WR_UNLOCK]. To protect programming same OTP bit twice, before program OCOTP will automatically read fuse value in OTP and use read value to mask program data. The controller will use masked program data to program a 32-bit word in the OTP per the address in HW_OCOTP_CTRL[ADDR]. Bit fields with 1's will result in that OTP bit being programmed. Bit fields with 0's will be ignored. At the same time that the write is accepted, the controller makes an internal copy of HW_OCOTP_CTRL[ADDR] which cannot be updated until the next write sequence is initiated. This copy guarantees that erroneous writes to HW_OCOTP_CTRL[ADDR] will not affect an active write operation. It should also be noted that during the programming HW_OCOTP_DATA will shift right (with zero fill). This shifting is required to program the OTP serially. During the write operation, HW_OCOTP_DATA cannot be modified.
5. Once complete, the controller will clear BUSY. A write request to a protected or locked region will result in no OTP access and no setting of HW_OCOTP_CTRL[BUSY]. In addition HW_OCOTP_CTRL[ERROR] will be set. It must be cleared by software before any new write access can be issued.

It should be noted that write latencies to OTP are numbers of 10 micro-seconds per word. Write latencies will be based on amount of bit filed which is 1. For example : program half fuse bits in one word need 10us x 16.

For further details of OTP read/write operations see [eFUSE].

HW_OCOTP_CTRL[ERROR] will be set under the following conditions:

- A write is performed to a shadow register during a shadow reload (essentially, while HW_OCOTP_CTRL[RELOAD_SHADOWS] is set. In addition, the contents of the shadow register shall not be updated.
- A write is performed to a shadow register which has been locked.
- A read is performed to from a shadow register which has been read locked.
- A program is performed to a fuse word which has been locked.
- A read is performed to from a fuse word which has been read locked.

46.3.1.4 Write Postamble

Due to internal electrical characteristics of the OTP during writes, all OTP operations following a write must be separated by 2 us after the clearing of HW_OCOTP_CTRL_BUSY following the write. This guarantees programming voltages on-chip to reach a steady state when exiting a write sequence. This includes reads, shadow reloads, or other writes.

A recommended software sequence to meet the postamble requirements is as follows:

- Issue the write and poll for BUSY (as per Fuse Shadow Memory Footprint).
- Once BUSY is clear, use HW_DIGCTL_MICROSECONDS to wait 2 us.
- Perform the next OTP operation.

46.3.2 Fuse Shadow Memory Footprint

The OTP memory footprint is shown in the figure below. The registers are grouped by lock region. Their names correspond to the PIO register and fusemap names.

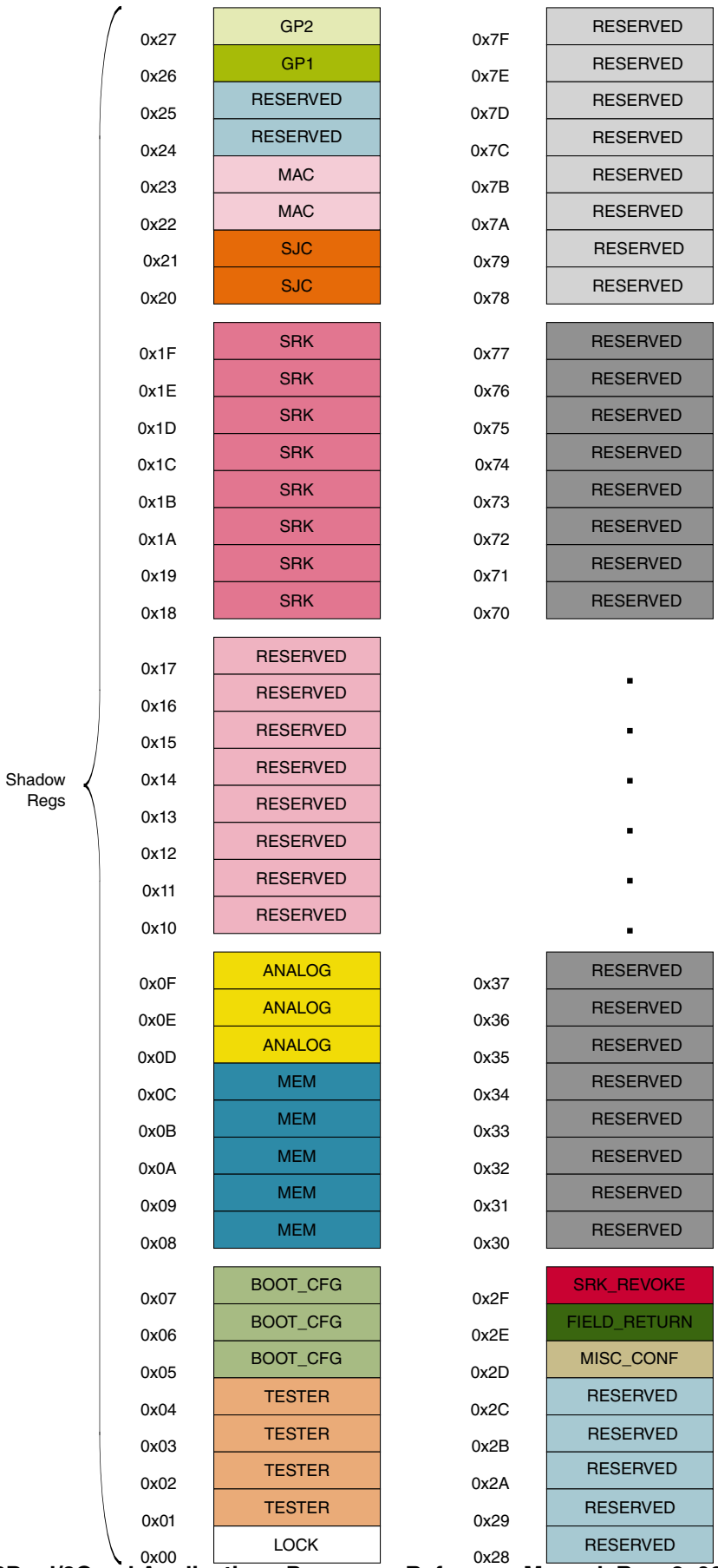


Figure 46-2. OTP Memory Footprint

46.3.3 OTP Read/Write Timing Parameters

There are 3 timing fields contained in the HW_OCOTP_TIMING register that specify counter limit values, which are used to time how long the state machine remains in the various states, as well as specify the STROBE signal timing.

They are all specified in ipg_clk cycles. Since the ipg_clk frequency can be set to a range of values, these parameters must be adjusted with the clock to yield the appropriate delay.

The HW_OCOTP_TIMING[RELAX] field specifies how long to remain in states to meet setup and hold timing requirement in fuse spec. This parameter should be set by the following equation:

tRELAX = tHP_PG = (HW_OCOTP_TIMING[RELAX]+1)/ipg_frequency > 16.2ns

HW_OCOTP_TIMING[RELAX] field is used to create other setup and hold timing delays in addition to tHP_PG. For all timing to be met, this is the max delay that must be programmed.

Except for setup and hold timing delay, there are 2 timing parameters for STROBE signal pulse width in program and read.

The HW_OCOTP_TIMING[STROBE_PROG] field specifies the period of the STROBE signal for fuse writes and is given in units of ipg_clk cycles. This value should be specified so that the requirement for the time when the STROBE signal is asserted high is met: 9000ns < tPGM < 11000ns is met. Even though a range is given for tPGM, it is advised in **[eFUSE]** to program for a value of 10000ns. Therefore, this field should be set according to the following equation:

tPGM = ((HW_OCOTP_TIMING[STROBE_PROG]+1)-
2*(HW_OCOTP_TIMING[RELAX]+1))/ipg_frequency = 10000ns.

The HW_OCOTP_TIMING[STROBE_READ] field specifies the period of the STROBE signal for fuse reads and is given in units of ipg_clk cycles. This field should be set according to the following equation:

tRD = ((HW_OCOTP_TIMING[STROBE_READ]+1)-
2*(HW_OCOTP_TIMING[RELAX]+1))/ipg_frequency > 36ns.

The figure below illustrates the relationship between the STROBE signal in programming and reading mode, as well as the timing PIO register fields that affect it. The implementation uses one counter to generate the STROBE waveform within one period and a second counter counts the number of cycles to create for programming the designated word.

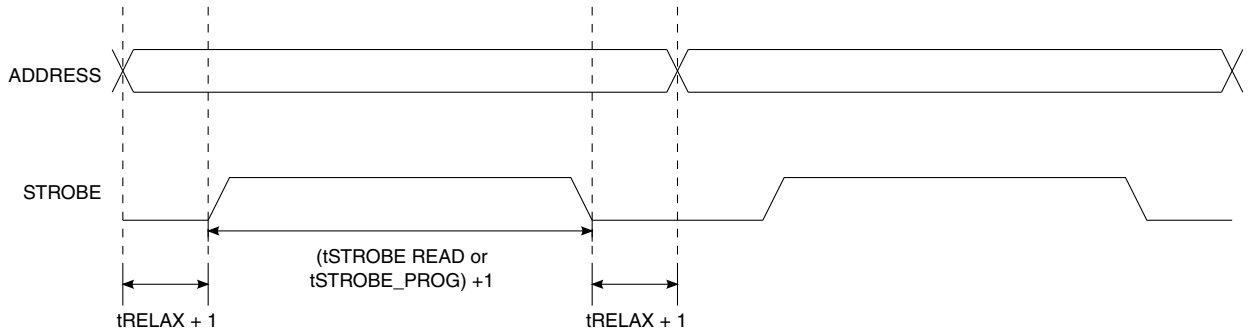


Figure 46-3. STROBE Signal Creation and Timing

46.3.4 Hardware Visible Fuses

The hwv_fuse bus emanates from the OCOTP block and goes to various other blocks inside the chip. This bus is made up of the shadow register bits for banks 0, 1, 2 and 4.

Only a subset of these fuse bits are currently used by the hardware. The fuse bits are initially copied from the eFuse banks after reset is deasserted. When all fuse bits are loaded into their shadow registers, the OCOTP asserts the fuse_latched output signal.

The hwv_reg bus also comes from the OCOTP. Its source is the HW_OCOTP_SCS register. This register has 1 defined bit, the HAB_JDE bit, that is connected to the SJC block. The SCS bits are intended to be used as volatile fuse bits under software control. Additional bits will be defined as needed in future implementations.

The system-wide reset sequence must be coordinated by the system reset controller so that the hwv_fuse and hwv_reg busses are stable and reflect the values of the fuses before they are used by the rest of the system.

46.3.5 Behavior During Reset

The OCOTP is always active. The shadow registers automatically load the appropriate OTP contents after reset is deasserted. During this load-time HW_OCOTP_CTRL_BUSY is set. The load time is similar to that of a "reload shadow" operation.

46.3.6 Secure JTAG control

The JTAG control fuses are used to allow or disallow JTAG access to secured resources.

Three JTAG security levels are envisioned, as shown in the table below.

Table 46-2. JTAG Security Level Control Bits

Security Mode	JTAG_SMODE	Description
No Debug	2'b11	The highest security level.
Secure JTAG	2'b01	Limit the JTAG access by using key based authentication mechanism.
JTAG Enable	2'b00	Low Security, all JTAG features are enabled.

46.4 Fuse Map

See the Fusemap chapter of this reference manual for more information.

46.5 OCOTP Memory Map/Register Definition

OCOTP Hardware Register Format Summary

OCOTP memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21B_C000	OTP Controller Control Register (OCOTP_CTRL)	32	R/W	0000_0000h	46.5.1/4032
21B_C004	OTP Controller Control Register (OCOTP_CTRL_SET)	32	R/W	0000_0000h	46.5.1/4032
21B_C008	OTP Controller Control Register (OCOTP_CTRL_CLR)	32	R/W	0000_0000h	46.5.1/4032
21B_C00C	OTP Controller Control Register (OCOTP_CTRL_TOG)	32	R/W	0000_0000h	46.5.1/4032
21B_C010	OTP Controller Timing Register (OCOTP_TIMING)	32	R/W	0146_1299h	46.5.2/4034
21B_C020	OTP Controller Write Data Register (OCOTP_DATA)	32	R/W	0000_0000h	46.5.3/4035
21B_C030	OTP Controller Write Data Register (OCOTP_READ_CTRL)	32	R/W	0000_0000h	46.5.4/4035
21B_C040	OTP Controller Read Data Register (OCOTP_READ_FUSE_DATA)	32	R/W	0000_0000h	46.5.5/4036
21B_C050	Sticky bit Register (OCOTP_SW_STICKY)	32	R/W	0000_0000h	46.5.6/4037
21B_C060	Software Controllable Signals Register (OCOTP_SCS)	32	R/W	0000_0000h	46.5.7/4038
21B_C064	Software Controllable Signals Register (OCOTP_SCS_SET)	32	R/W	0000_0000h	46.5.7/4038
21B_C068	Software Controllable Signals Register (OCOTP_SCS_CLR)	32	R/W	0000_0000h	46.5.7/4038
21B_C06C	Software Controllable Signals Register (OCOTP_SCS_TOG)	32	R/W	0000_0000h	46.5.7/4038
21B_C090	OTP Controller Version Register (OCOTP_VERSION)	32	R	0200_0000h	46.5.8/4039

Table continues on the next page...

OCOTP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21B_C400	Value of OTP Bank0 Word0 (Lock controls) (OCOTP_LOCK)	32	R	0000_0000h	46.5.9/4039
21B_C410	Value of OTP Bank0 Word1 (Configuration and Manufacturing Info.) (OCOTP_CFG0)	32	R/W	0000_0000h	46.5.10/ 4042
21B_C420	Value of OTP Bank0 Word2 (Configuration and Manufacturing Info.) (OCOTP_CFG1)	32	R/W	0000_0000h	46.5.11/ 4043
21B_C430	Value of OTP Bank0 Word3 (Configuration and Manufacturing Info.) (OCOTP_CFG2)	32	R/W	0000_0000h	46.5.12/ 4043
21B_C440	Value of OTP Bank0 Word4 (Configuration and Manufacturing Info.) (OCOTP_CFG3)	32	R/W	0000_0000h	46.5.13/ 4044
21B_C450	Value of OTP Bank0 Word5 (Configuration and Manufacturing Info.) (OCOTP_CFG4)	32	R/W	0000_0000h	46.5.14/ 4044
21B_C460	Value of OTP Bank0 Word6 (Configuration and Manufacturing Info.) (OCOTP_CFG5)	32	R/W	0000_0000h	46.5.15/ 4045
21B_C470	Value of OTP Bank0 Word7 (Configuration and Manufacturing Info.) (OCOTP_CFG6)	32	R/W	0000_0000h	46.5.16/ 4045
21B_C480	Value of OTP Bank1 Word0 (Memory Related Info.) (OCOTP_MEM0)	32	R/W	0000_0000h	46.5.17/ 4046
21B_C490	Value of OTP Bank1 Word1 (Memory Related Info.) (OCOTP_MEM1)	32	R/W	0000_0000h	46.5.18/ 4046
21B_C4A0	Value of OTP Bank1 Word2 (Memory Related Info.) (OCOTP_MEM2)	32	R/W	0000_0000h	46.5.19/ 4047
21B_C4B0	Value of OTP Bank1 Word3 (Memory Related Info.) (OCOTP_MEM3)	32	R/W	0000_0000h	46.5.20/ 4047
21B_C4C0	Value of OTP Bank1 Word4 (Memory Related Info.) (OCOTP_MEM4)	32	R/W	0000_0000h	46.5.21/ 4048
21B_C4D0	Value of OTP Bank1 Word5 (Memory Related Info.) (OCOTP_ANA0)	32	R/W	0000_0000h	46.5.22/ 4048
21B_C4E0	Value of OTP Bank1 Word6 (General Purpose Customer Defined Info.) (OCOTP_ANA1)	32	R/W	0000_0000h	46.5.23/ 4049
21B_C4F0	Value of OTP Bank1 Word7 (General Purpose Customer Defined Info.) (OCOTP_ANA2)	32	R/W	0000_0000h	46.5.24/ 4049
21B_C580	Shadow Register for OTP Bank3 Word0 (SRK Hash) (OCOTP_SRK0)	32	R/W	0000_0000h	46.5.25/ 4050
21B_C590	Shadow Register for OTP Bank3 Word1 (SRK Hash) (OCOTP_SRK1)	32	R/W	0000_0000h	46.5.26/ 4050
21B_C5A0	Shadow Register for OTP Bank3 Word2 (SRK Hash) (OCOTP_SRK2)	32	R/W	0000_0000h	46.5.27/ 4051
21B_C5B0	Shadow Register for OTP Bank3 Word3 (SRK Hash) (OCOTP_SRK3)	32	R/W	0000_0000h	46.5.28/ 4051
21B_C5C0	Shadow Register for OTP Bank3 Word4 (SRK Hash) (OCOTP_SRK4)	32	R/W	0000_0000h	46.5.29/ 4052
21B_C5D0	Shadow Register for OTP Bank3 Word5 (SRK Hash) (OCOTP_SRK5)	32	R/W	0000_0000h	46.5.30/ 4052

Table continues on the next page...

OCOTP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21B_C5E0	Shadow Register for OTP Bank3 Word6 (SRK Hash) (OCOTP_SRK6)	32	R/W	0000_0000h	46.5.31/ 4053
21B_C5F0	Shadow Register for OTP Bank3 Word7 (SRK Hash) (OCOTP_SRK7)	32	R/W	0000_0000h	46.5.32/ 4053
21B_C600	Value of OTP Bank4 Word0 (Secure JTAG Response Field) (OCOTP_RESP0)	32	R/W	0000_0000h	46.5.33/ 4054
21B_C610	Value of OTP Bank4 Word1 (Secure JTAG Response Field) (OCOTP_HSJC_RESP1)	32	R/W	0000_0000h	46.5.34/ 4054
21B_C620	Value of OTP Bank4 Word2 (MAC Address) (OCOTP_MAC0)	32	R/W	0000_0000h	46.5.35/ 4055
21B_C630	Value of OTP Bank4 Word3 (MAC Address) (OCOTP_MAC1)	32	R/W	0000_0000h	46.5.36/ 4055
21B_C660	Value of OTP Bank4 Word6 (HW Capabilities) (OCOTP_GP1)	32	R/W	0000_0000h	46.5.37/ 4056
21B_C670	Value of OTP Bank4 Word7 (HW Capabilities) (OCOTP_GP2)	32	R/W	0000_0000h	46.5.38/ 4056
21B_C6D0	Value of OTP Bank5 Word5 (HW Capabilities) (OCOTP_MISC_CONF)	32	R/W	0000_0000h	46.5.39/ 4057
21B_C6E0	Value of OTP Bank5 Word6 (HW Capabilities) (OCOTP_FIELD_RETURN)	32	R/W	0000_0000h	46.5.40/ 4057
21B_C6F0	Value of OTP Bank5 Word7 (HW Capabilities) (OCOTP_SRK_REVOKE)	32	R/W	0000_0000h	46.5.41/ 4058

46.5.1 OTP Controller Control Register (OCOTP_CTRL*n*)

The OCOTP Control and Status Register specifies the copy state, as well as the control required for random access of the OTP memory

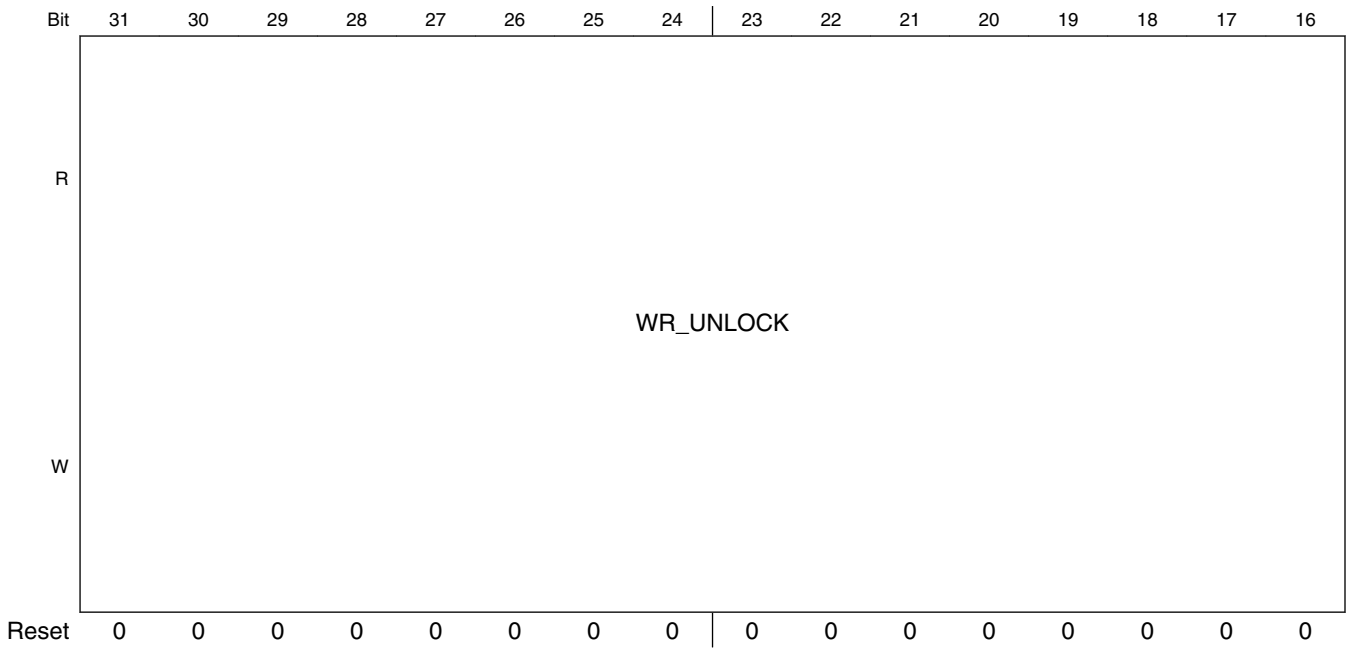
OCOTP_CTRL: 0x000

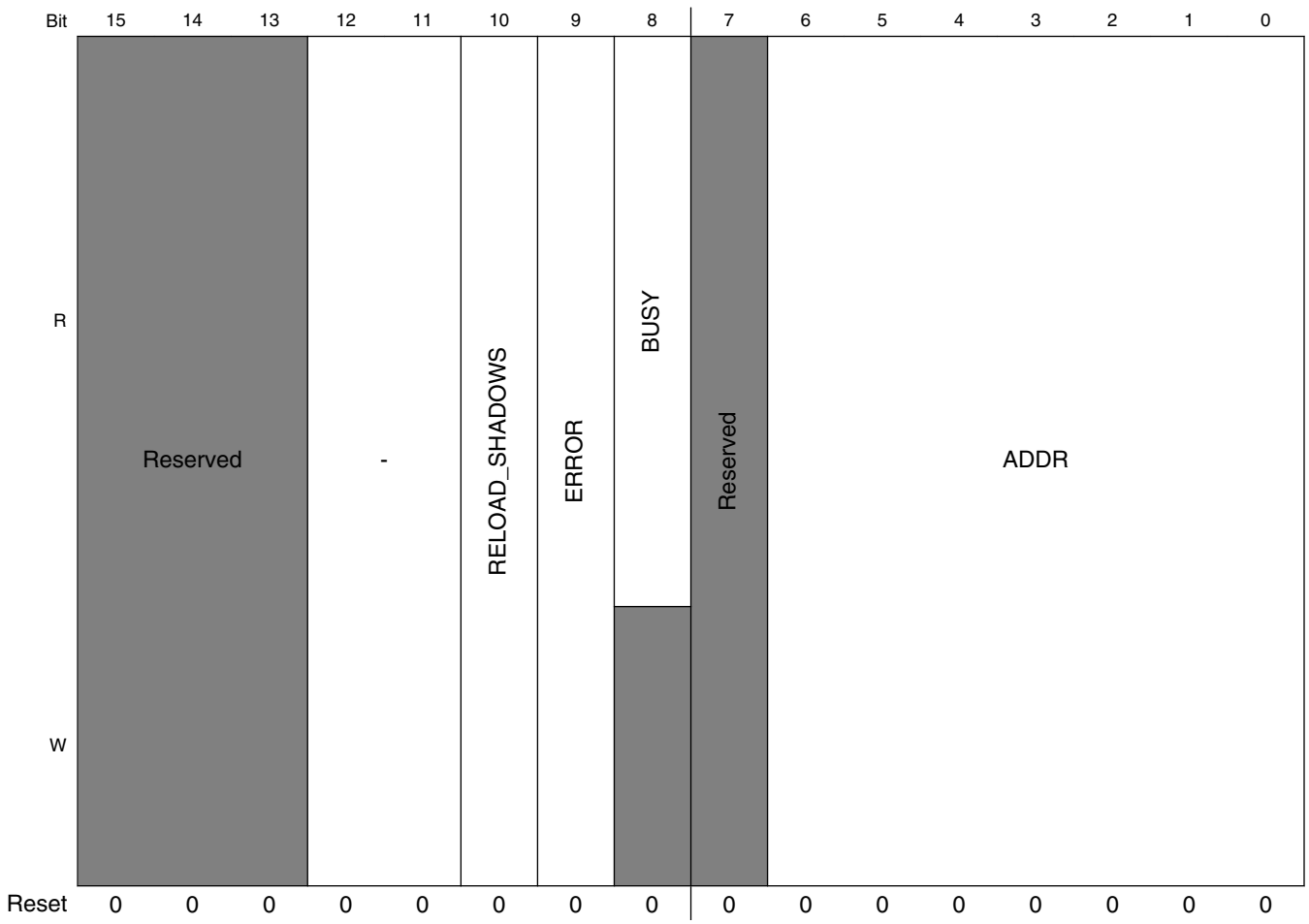
The OCOTP Control and Status Register provides the necessary software interface for performing read and write operations to the On-Chip OTP (One-Time Programmable ROM). The control fields such as WR_UNLOCK, ADDR and BUSY/ERROR may be used in conjuction with the HW_OCOTP_DATA register to perform write operations. Read operations to the On-Chip OTP are involving ADDR, BUSY/ERROR bit field and HW_OCOTP_READ_CTRL register. Read value is saved in HW_OCOTP_READ_FUSE_DATA register.

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 0h offset + (4d × i), where i=0d to 3d





OCOTP_CTRL*n* field descriptions

Field	Description
31–16 WR_UNLOCK	Write 0x3E77 to enable OTP write accesses. NOTE: This register must be unlocked on a write-by-write basis (a write is initiated when HW_OCOTP_DATA is written), so the UNLOCK bitfield must contain the correct key value during all writes to HW_OCOTP_DATA, otherwise a write shall not be initiated. This field is automatically cleared after a successful write completion (clearing of BUSY). 0x3E77 **KEY** — Key needed to unlock HW_OCOTP_DATA register.
15–13 -	This field is reserved. Reserved
12–11 -	Reserved
10 RELOAD_ SHADOWS	Set to force re-loading the shadow registers (HW/SW capability and LOCK). This operation will automatically set BUSY. Once the shadow registers have been re-loaded, BUSY and RELOAD_SHADOWS are automatically cleared by the controller.
9 ERROR	Set by the controller when an access to a locked region(OTP or shadow register) is requested. Must be cleared before any further access can be performed. This bit can only be set by the controller. This bit is also set if the Pin interface is active and software requests an access to the OTP. In this instance, the ERROR bit cannot be cleared until the Pin interface access has completed. Reset this bit by writing a one to the SCT clear address space and not by a general write.
8 BUSY	OTP controller status bit. When active, no new write access or read access to OTP(including RELOAD_SHADOWS) can be performed. Cleared by controller when access complete. After reset (or after setting RELOAD_SHADOWS), this bit is set by the controller until the HW/SW and LOCK registers are successfully copied, after which time it is automatically cleared by the controller.
7 -	This field is reserved. Reserved
ADDR	OTP write and read access address register. Specifies one of 128 word address locations (0x00 - 0x7f). If a valid access is accepted by the controller, the controller makes an internal copy of this value. This internal copy will not update until the access is complete.

46.5.2 OTP Controller Timing Register (OCOTP_TIMING)

The OCOTP Data Register is used for OTP Programming

This register specifies timing parameters for programming and reading the OCOTP fuse array.

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 10h offset = 21B_C010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				WAIT						STROBE_READ						RELAX				STROBE_PROG											
W																																
Reset	0	0	0	0	0	0	0	1	0	1	0	0	0	1	1	0	0	0	0	1	0	0	1	0	1	0	0	1	1	0	0	1

OCOTP_TIMING field descriptions

Field	Description
31–28 Reserved	This read-only field is reserved and always has the value 0.
27–22 WAIT	This count value specifies time interval between auto read and write access in one time program. It is given in number of ipg_clk periods.
21–16 STROBE_READ	This count value specifies the strobe period in one time read OTP. Trd = ((STROBE_READ+1)- 2*(RELAX +1)) /ipg_clk_freq. It is given in number of ipg_clk periods.
15–12 RELAX	This count value specifies the time to add to all default timing parameters other than the Tpgm and Trd. It is given in number of ipg_clk periods.
STROBE_PROG	This count value specifies the strobe period in one time write OTP. Tpgm = ((STROBE_PROG+1)- 2*(RELAX+1)) /ipg_clk_freq. It is given in number of ipg_clk periods.

46.5.3 OTP Controller Write Data Register (OCOTP_DATA)

The OCOTP Data Register is used for OTP Programming

This register is used in conjuction with HW_OCOTP_CTRL to perform one-time writes to the OTP. Please see the "Software Write Sequence" section for operating details.

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 20h offset = 21B_C020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_DATA field descriptions

Field	Description
DATA	Used to initiate a write to OTP. Please see the "Software Write Sequence" section for operating details.

46.5.4 OTP Controller Write Data Register (OCOTP_READ_CTRL)

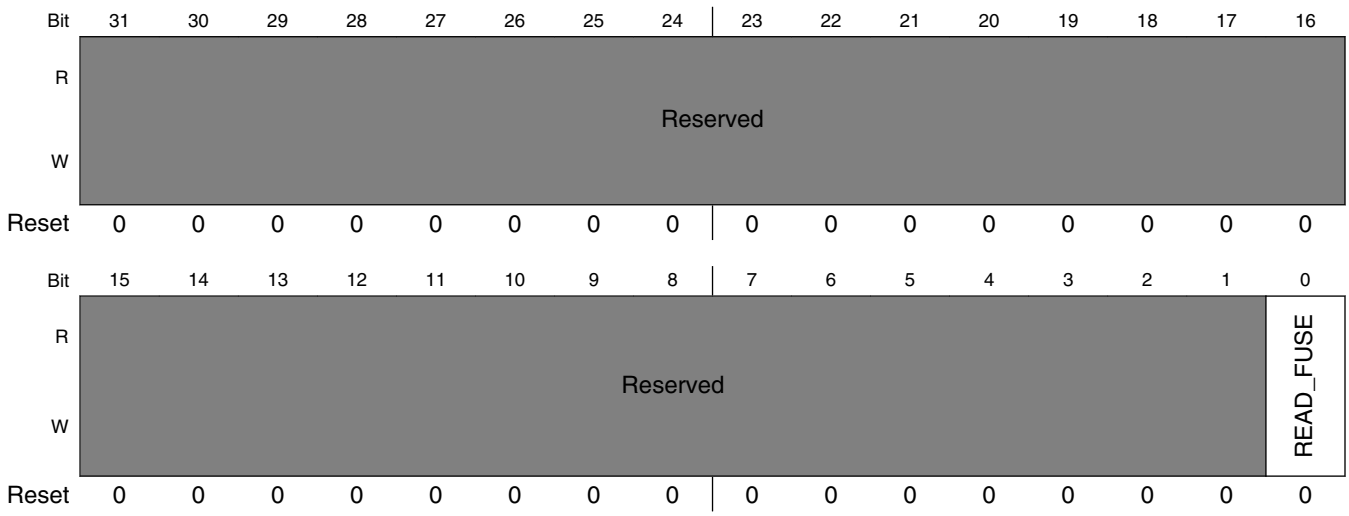
The OCOTP Register is used for OTP Read

This register is used in conjuction with HW_OCOTP_CTRL to perform one time read to the OTP. Please see the "Software read Sequence" section for operating details.

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 30h offset = 21B_C030h



OCOTP_READ_CTRL field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 READ_FUSE	Used to initiate a read to OTP. Please see the "Software read Sequence" section for operating details.

46.5.5 OTP Controller Read Data Register (OCOTP_READ_FUSE_DATA)

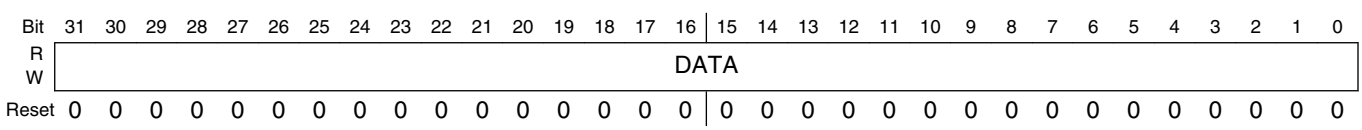
The OCOTP Data Register is used for OTP Read

The data read from OTP

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 40h offset = 21B_C040h



OCOTP_READ_FUSE_DATA field descriptions

Field	Description
DATA	The data read from OTP

46.5.6 Sticky bit Register (OCOTP_SW_STICKY)

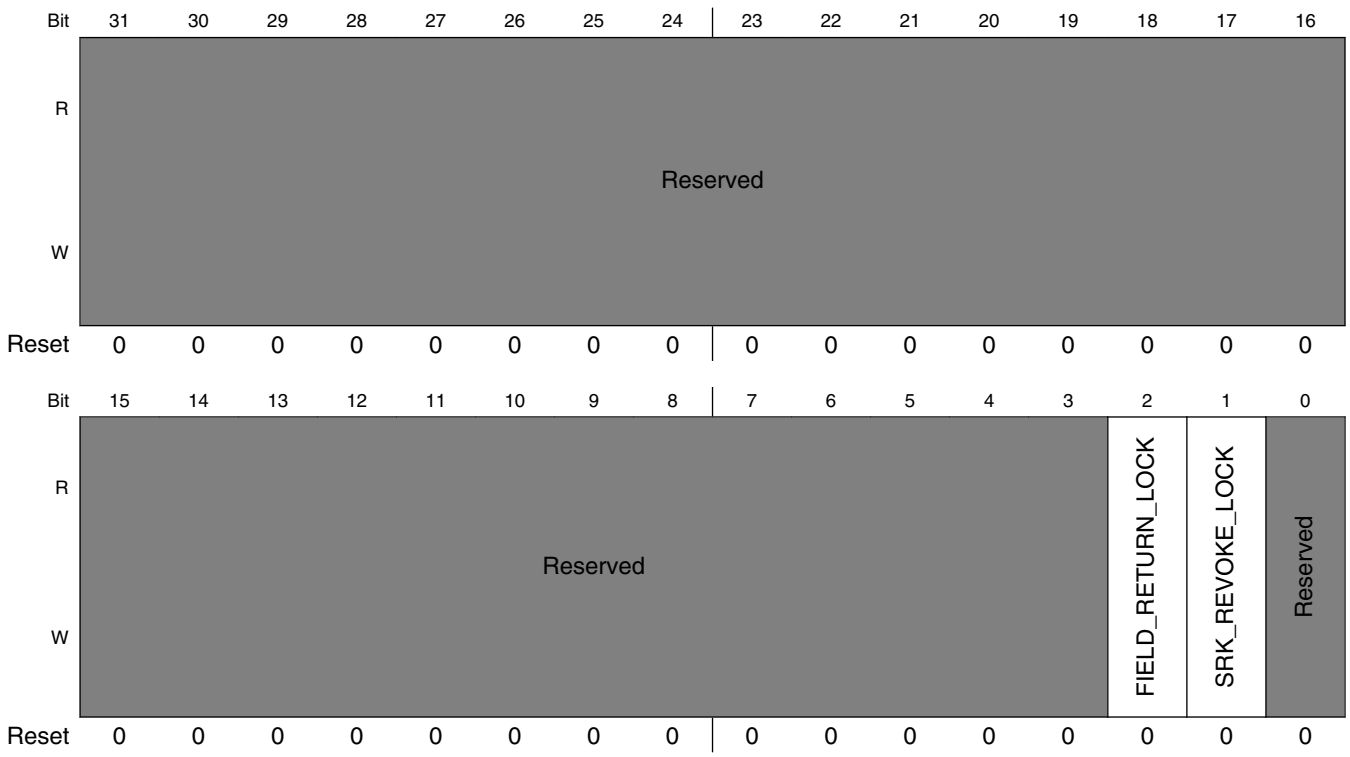
Some SW sticky bits .

Some sticky bits are used by SW to lock some fuse area , shadow registers and other features.

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 50h offset = 21B_C050h



OCOTP_SW_STICKY field descriptions

Field	Description
31–3 -	This field is reserved. Reserved
2 FIELD_ RETURN_LOCK	Shadow register write and OTP write lock for FIELD_RETURN region. When set, the writing of this region's shadow register and OTP fuse word are blocked.Once this bit is set, it is always high unless a POR is issued.
1 SRK_REVOKE_ LOCK	Shadow register write and OTP write lock for SRK_REVOKE, MC_ERA and AP_BI_VER regions. When set, the writing of these region's shadow register and OTP fuse word are blocked. Once this bit is set, it is always high unless a POR is issued.
0 -	This field is reserved. Reserved.

46.5.7 Software Controllable Signals Register (OCOTP_SCS*n*)

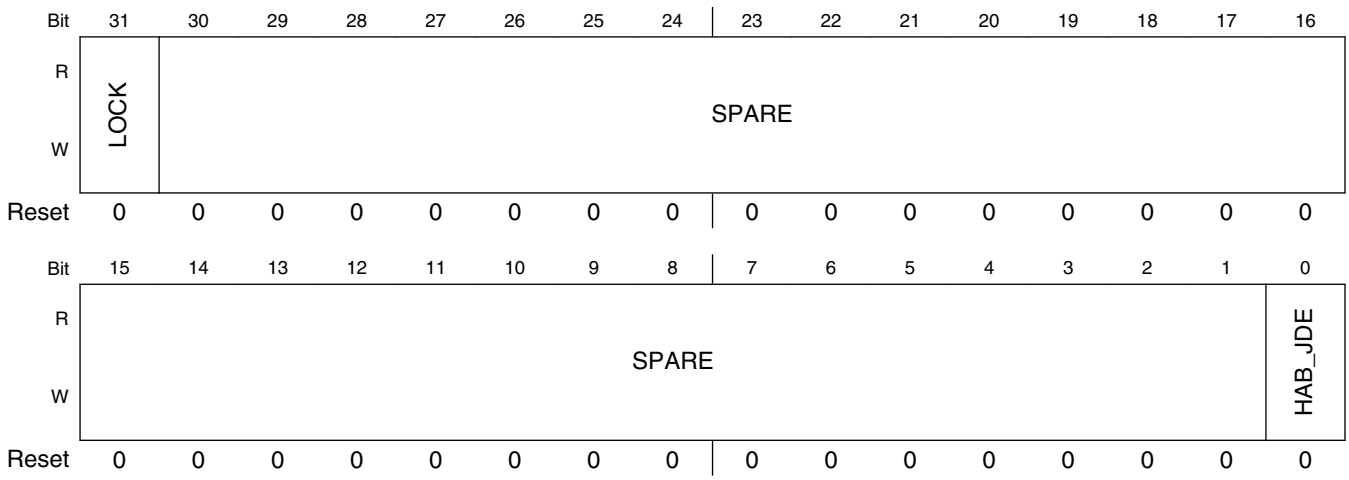
HW_OCOTP_SCS: 0x060

This register holds volatile configuration values that can be set and locked by trusted software. All values are returned to their defualt values after POR.

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 60h offset + (4d × i), where i=0d to 3d



OCOTP_SCS*n* field descriptions

Field	Description
31 LOCK	When set, all of the bits in this register are locked and can not be changed through SW programming. This bit is only reset after a POR is issued.
30–1 SPARE	Unallocated read/write bits for implementation specific software use.
0 HAB_JDE	HAB JTAG Debug Enable. This bit is used by the HAB to enable JTAG debugging, assuming that a properly signed command to do so is found and validated by the HAB. The HAB must lock the register before passing control to the OS whether or not JTAG debugging has been enabled. Once JTAG is enabled by this bit, it can not be disabled unless the system is reset by POR. 0: JTAG debugging is not enabled by the HAB (it may still be enabled by other mechanisms). 1: JTAG debugging is enabled by the HAB (though this signal may be gated off). 1 JTAG debugging is enabled by the HAB (though this signal may be gated off)

46.5.8 OTP Controller Version Register (OCOTP_VERSION)

This register always returns a known read value for debug purposes it indicates the version of the block.

This register indicates the RTL version in use.

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 90h offset = 21B_C090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAJOR								MINOR								STEP															
W																																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.
STEP	Fixed read-only value reflecting the stepping of the RTL version.

46.5.9 Value of OTP Bank0 Word0 (Lock controls) (OCOTP_LOCK)

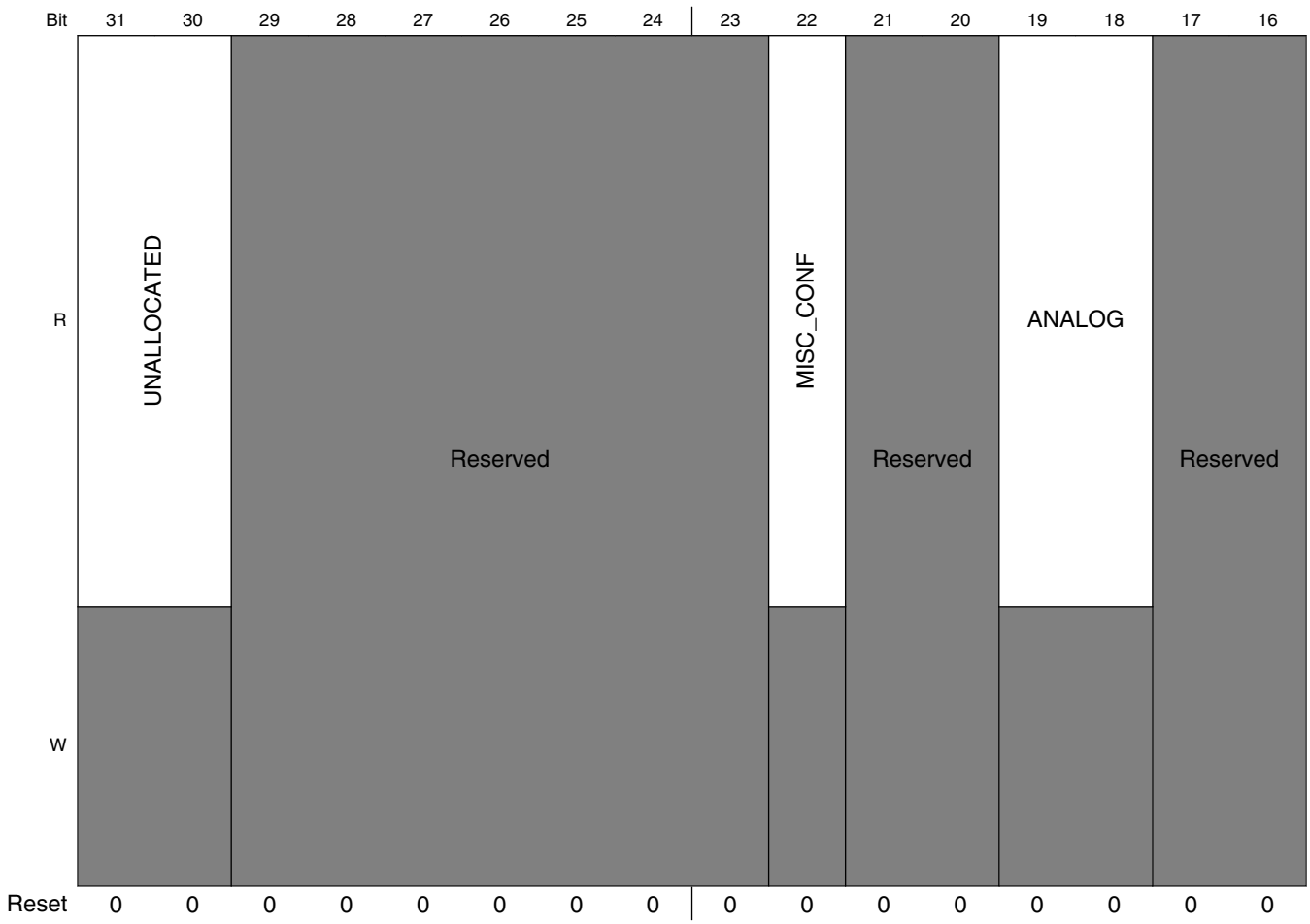
Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

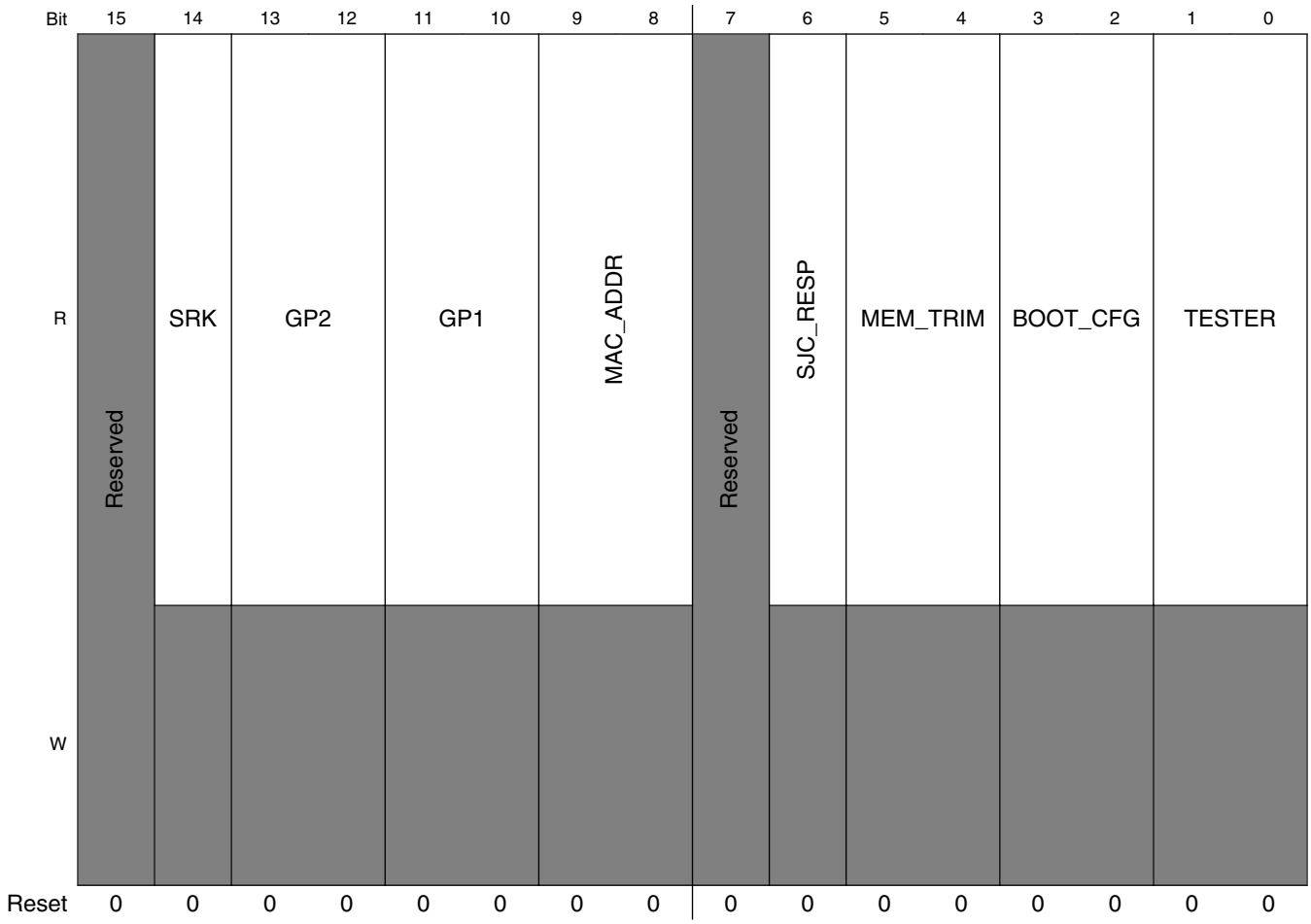
Shadowed memory mapped access to OTP Bank 0, word 0 (ADDR = 0x00).

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 400h offset = 21B_C400h





OCOTP_LOCK field descriptions

Field	Description
31–30 UNALLOCATED	Value of un-used portion of LOCK word
29–23 -	This field is reserved. Reserved
22 MISC_CONF	Status of shadow register and OTP write lock for misc_conf region. When set, the writing of this region's shadow register and OTP fuse word are blocked.
21–20 -	This field is reserved. Reserved
19–18 ANALOG	Status of shadow register and OTP write lock for analog region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.
17–16 -	This field is reserved.
15 -	This field is reserved. Reserved
14 SRK	Status of shadow register and OTP write lock for srk region. When set, the writing of this region's shadow register and OTP fuse word are blocked.

Table continues on the next page...

OCOTP_LOCK field descriptions (continued)

Field	Description
13–12 GP2	Status of shadow register and OTP write lock for gp2 region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.
11–10 GP1	Status of shadow register and OTP write lock for gp2 region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.
9–8 MAC_ADDR	Status of shadow register and OTP write lock for mac_addr region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.
7 -	This field is reserved. Reserved
6 SJC_RESP	Status of shadow register read and write, OTP read and write lock for sjc_resp region. When set, the writing of this region's shadow register and OTP fuse word are blocked. The read of this region's shadow register and OTP fuse word are also blocked.
5–4 MEM_TRIM	Status of shadow register and OTP write lock for mem_trim region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.
3–2 BOOT_CFG	Status of shadow register and OTP write lock for boot_cfg region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.
TESTER	Status of shadow register and OTP write lock for tester region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.

46.5.10 Value of OTP Bank0 Word1 (Configuration and Manufacturing Info.) (OCOTP_CFG0)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 0, word 1 (ADDR = 0x01).

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 410h offset = 21B_C410h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_CFG0 field descriptions

Field	Description
BITS	This register contains 32 bits of the Unique ID and SJC_CHALLENGE field. Reflects value of OTP Bank 0, word 1 (ADDR = 0x01). These bits become read-only after the HW_OCOTP_LOCK_TESTER[1] bit is set.

46.5.11 Value of OTP Bank0 Word2 (Configuration and Manufacturing Info.) (OCOTP_CFG1)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

shadowed memory mapped access to OTP Bank 0, word 2 (ADDR = 0x02).

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 420h offset = 21B_C420h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_CFG1 field descriptions

Field	Description
BITS	This register contains 32 bits of the Unique ID and SJC_CHALLENGE field. Reflects value of OTP Bank 0, word 2 (ADDR = 0x02). These bits become read-only after the HW_OCOTP_LOCK_TESTER[1] bit is set.

46.5.12 Value of OTP Bank0 Word3 (Configuration and Manufacturing Info.) (OCOTP_CFG2)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 0, word 3 (ADDR = 0x03).

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 430h offset = 21B_C430h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_CFG2 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 0, word 3 (ADDR = 0x03). These bits become read-only after the HW_OCOTP_LOCK_TESTER[1] bit is set.

46.5.13 Value of OTP Bank0 Word4 (Configuration and Manufacturing Info.) (OCOTP_CFG3)

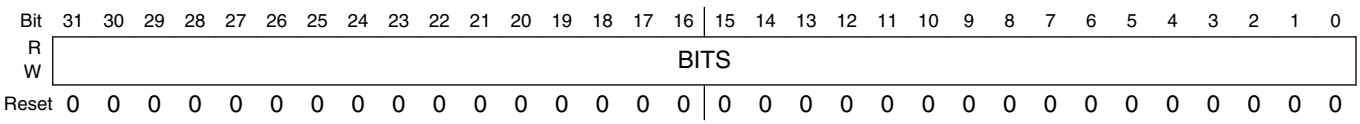
Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Non-shadowed memory mapped access to OTP Bank 0, word 4 (ADDR = 0x04).

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 440h offset = 21B_C440h



OCOTP_CFG3 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 0, word 4 (ADDR = 0x04). These bits become read-only after the HW_OCOTP_LOCK_TESTER[1] bit is set.

46.5.14 Value of OTP Bank0 Word5 (Configuration and Manufacturing Info.) (OCOTP_CFG4)

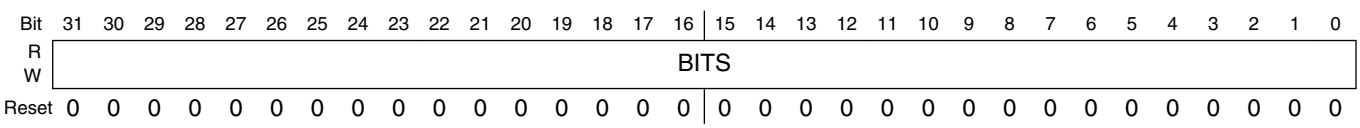
Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 0, word 5 (ADDR = 0x05).

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 450h offset = 21B_C450h



OCOTP_CFG4 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 0, word 5 (ADDR = 0x05). These bits become read-only after the HW_OCOTP_LOCK_BOOT_CFG[1] bit is set.

46.5.15 Value of OTP Bank0 Word6 (Configuration and Manufacturing Info.) (OCOTP_CFG5)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 0, word 6 (ADDR = 0x06).

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 460h offset = 21B_C460h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_CFG5 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 0, word 6 (ADDR = 0x06). These bits become read-only after the HW_OCOTP_LOCK_BOOT_CFG[1] bit is set.

46.5.16 Value of OTP Bank0 Word7 (Configuration and Manufacturing Info.) (OCOTP_CFG6)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 0, word 7 (ADDR = 0x07).

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 470h offset = 21B_C470h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_CFG6 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 0, word 7 (ADDR = 0x07). These bits become read-only after the HW_OCOTP_LOCK_BOOT_CFG[1] bit is set.

46.5.17 Value of OTP Bank1 Word0 (Memory Related Info.) (OCOTP_MEM0)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 1, word 0 (ADDR = 0x08).

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 480h offset = 21B_C480h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_MEM0 field descriptions

Field	Description
BITS	Reflects value of OTP bank 1, word 0 (ADDR = 0x08). These bits become read-only after the HW_OCOTP_LOCK_MEM_TRIM[1] bit is set.

46.5.18 Value of OTP Bank1 Word1 (Memory Related Info.) (OCOTP_MEM1)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 1, word 1 (ADDR = 0x09).

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 490h offset = 21B_C490h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_MEM1 field descriptions

Field	Description
BITS	Reflects value of OTP bank 1, word 1 (ADDR = 0x09). These bits become read-only after the HW_OCOTP_LOCK_MEM_TRIM[1] bit is set.

46.5.19 Value of OTP Bank1 Word2 (Memory Related Info.) (OCOTP_MEM2)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 1, word 2 (ADDR = 0x0A).

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 4A0h offset = 21B_C4A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_MEM2 field descriptions

Field	Description
BITS	Reflects value of OTP bank 1, word 2 (ADDR = 0x0A). These bits become read-only after the HW_OCOTP_LOCK_MEM_TRIM[1] bit is set.

46.5.20 Value of OTP Bank1 Word3 (Memory Related Info.) (OCOTP_MEM3)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 1, word 3 (ADDR = 0x0B).

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 4B0h offset = 21B_C4B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_MEM3 field descriptions

Field	Description
BITS	Reflects value of OTP bank 1, word 3 (ADDR = 0x0B). These bits become read-only after the HW_OCOTP_LOCK_MEM_TRIM[1] bit is set.

46.5.21 Value of OTP Bank1 Word4 (Memory Related Info.) (OCOTP_MEM4)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 1, word 4 (ADDR = 0x0C).

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 4C0h offset = 21B_C4C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_MEM4 field descriptions

Field	Description
BITS	Reflects value of OTP bank 1, word 4 (ADDR = 0x0C). These bits become read-only after the HW_OCOTP_LOCK_MEM_TRIM[1] bit is set.

46.5.22 Value of OTP Bank1 Word5 (Memory Related Info.) (OCOTP_ANA0)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 1, word 5 (ADDR = 0x0D).

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 4D0h offset = 21B_C4D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_ANA0 field descriptions

Field	Description
BITS	Reflects value of OTP bank 1, word 5 (ADDR = 0x0D). These bits become read-only after the HW_OCOTP_LOCK_ANALOG[1] bit is set.

46.5.23 Value of OTP Bank1 Word6 (General Purpose Customer Defined Info.) (OCOTP_ANA1)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 1, word 6 (ADDR = 0x0E).

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 4E0h offset = 21B_C4E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_ANA1 field descriptions

Field	Description
BITS	Reflects value of OTP bank 1, word 6 (ADDR = 0x0E). These bits become read-only after the HW_OCOTP_LOCK_ANALOG[1] bit is set.

46.5.24 Value of OTP Bank1 Word7 (General Purpose Customer Defined Info.) (OCOTP_ANA2)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 1, word 7 (ADDR = 0x0F).

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 4F0h offset = 21B_C4F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_ANA2 field descriptions

Field	Description
BITS	Reflects value of OTP bank 1, word 7 (ADDR = 0x0F). These bits become read-only after the HW_OCOTP_LOCK_ANALOG[1] bit is set.

46.5.25 Shadow Register for OTP Bank3 Word0 (SRK Hash) (OCOTP_SRK0)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 3, word 0 (ADDR = 0x18).

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 580h offset = 21B_C580h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_SRK0 field descriptions

Field	Description
BITS	Shadow register for the hash of the Super Root Key word0 (Copy of OTP Bank 3, word 0 (ADDR = 0x1C)). These bits become read-only after the HW_OCOTP_LOCK_SRK bit is set.

46.5.26 Shadow Register for OTP Bank3 Word1 (SRK Hash) (OCOTP_SRK1)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 3, word 1 (ADDR = 0x19).

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 590h offset = 21B_C590h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_SRK1 field descriptions

Field	Description
BITS	Shadow register for the hash of the Super Root Key word1 (Copy of OTP Bank 3, word 1 (ADDR = 0x1D)). These bits become read-only after the HW_OCOTP_LOCK_SRK bit is set.

46.5.27 Shadow Register for OTP Bank3 Word2 (SRK Hash) (OCOTP_SRK2)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 3, word 2 (ADDR = 0x1A).

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 5A0h offset = 21B_C5A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_SRK2 field descriptions

Field	Description
BITS	Shadow register for the hash of the Super Root Key word2 (Copy of OTP Bank 3, word 2 (ADDR = 0x1E)). These bits become read-only after the HW_OCOTP_LOCK_SRK bit is set.

46.5.28 Shadow Register for OTP Bank3 Word3 (SRK Hash) (OCOTP_SRK3)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 3, word 3 (ADDR = 0x1B).

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 5B0h offset = 21B_C5B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_SRK3 field descriptions

Field	Description
BITS	Shadow register for the hash of the Super Root Key word3 (Copy of OTP Bank 3, word 3 (ADDR = 0x1F)). These bits become read-only after the HW_OCOTP_LOCK_SRK bit is set.

46.5.29 Shadow Register for OTP Bank3 Word4 (SRK Hash) (OCOTP_SRK4)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 3, word 4 (ADDR = 0x1C).

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 5C0h offset = 21B_C5C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_SRK4 field descriptions

Field	Description
BITS	Shadow register for the hash of the Super Root Key word4 (Copy of OTP Bank 3, word 4 (ADDR = 0x20)). These bits become read-only after the HW_OCOTP_LOCK_SRK bit is set.

46.5.30 Shadow Register for OTP Bank3 Word5 (SRK Hash) (OCOTP_SRK5)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 3, word 5 (ADDR = 0x1D).

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 5D0h offset = 21B_C5D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_SRK5 field descriptions

Field	Description
BITS	Shadow register for the hash of the Super Root Key word5 (Copy of OTP Bank 3, word 5 (ADDR = 0x21)). These bits become read-only after the HW_OCOTP_LOCK_SRK bit is set.

46.5.31 Shadow Register for OTP Bank3 Word6 (SRK Hash) (OCOTP_SRK6)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 3, word 6 (ADDR = 0x1E).

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 5E0h offset = 21B_C5E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_SRK6 field descriptions

Field	Description
BITS	Shadow register for the hash of the Super Root Key word6 (Copy of OTP Bank 3, word 6 (ADDR = 0x22)). These bits become read-only after the HW_OCOTP_LOCK_SRK bit is set.

46.5.32 Shadow Register for OTP Bank3 Word7 (SRK Hash) (OCOTP_SRK7)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 3, word 7 (ADDR = 0x1F).

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 5F0h offset = 21B_C5F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_SRK7 field descriptions

Field	Description
BITS	Shadow register for the hash of the Super Root Key word7 (Copy of OTP Bank 3, word 7 (ADDR = 0x23)). These bits become read-only after the HW_OCOTP_LOCK_SRK bit is set.

46.5.33 Value of OTP Bank4 Word0 (Secure JTAG Response Field) (OCOTP_RESP0)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 4, word 0 (ADDR = 0x20).

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 600h offset = 21B_C600h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_RESP0 field descriptions

Field	Description
BITS	Shadow register for the SJC_RESP Key word0 (Copy of OTP Bank 4, word 0 (ADDR = 0x20)). These bits can be not read and written after the HW_OCOTP_LOCK_SJC_RESP bit is set. If read, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR].

46.5.34 Value of OTP Bank4 Word1 (Secure JTAG Response Field) (OCOTP_HSJC_RESP1)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 4, word 1 (ADDR = 0x21).

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 610h offset = 21B_C610h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_HSJC_RESP1 field descriptions

Field	Description
BITS	Shadow register for the SJC_RESP Key word1 (Copy of OTP Bank 4, word 1 (ADDR = 0x21)). These bits can be not read and written after the HW_OCOTP_LOCK_SJC_RESP bit is set. If read, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR].

46.5.35 Value of OTP Bank4 Word2 (MAC Address) (OCOTP_MAC0)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 4, word 2 (ADDR = 0x22).

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 620h offset = 21B_C620h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_MAC0 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 4, word 2 (ADDR = 0x22).

46.5.36 Value of OTP Bank4 Word3 (MAC Address) (OCOTP_MAC1)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 4, word 3 (ADDR = 0x23).

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 630h offset = 21B_C630h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_MAC1 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 4, word 3 (ADDR = 0x23).

46.5.37 Value of OTP Bank4 Word6 (HW Capabilities) (OCOTP_GP1)

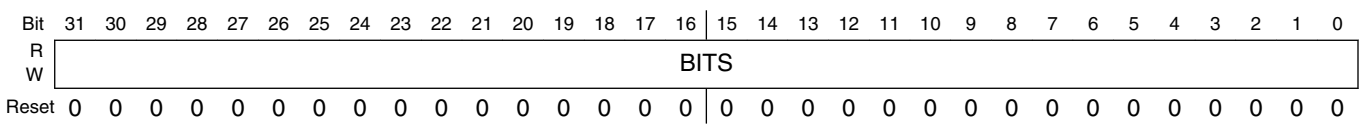
Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 4, word 6 (ADDR = 0x26).

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 660h offset = 21B_C660h



OCOTP_GP1 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 4, word 6 (ADDR = 0x26).

46.5.38 Value of OTP Bank4 Word7 (HW Capabilities) (OCOTP_GP2)

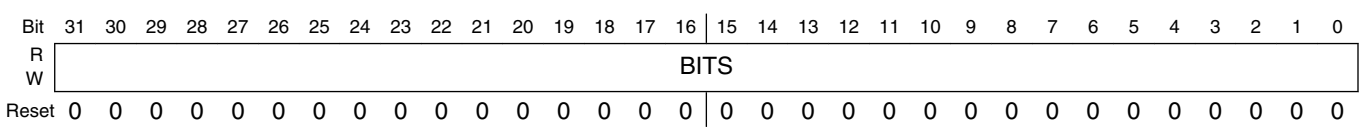
Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 4, word 7 (ADDR = 0x27).

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 670h offset = 21B_C670h



OCOTP_GP2 field descriptions

Field	Description
BITS	Reflects value of OTP Bank 4, word 7 (ADDR = 0x27).

46.5.39 Value of OTP Bank5 Word5 (HW Capabilities) (OCOTP_MISC_CONF)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 5, word 5 (ADDR = 0x2d).

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 6D0h offset = 21B_C6D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_MISC_CONF field descriptions

Field	Description
BITS	Reflects value of OTP Bank 5, word 5 (ADDR = 0x2d).

46.5.40 Value of OTP Bank5 Word6 (HW Capabilities) (OCOTP_FIELD_RETURN)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 5, word 6 (ADDR = 0x2e).

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 6E0h offset = 21B_C6E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_FIELD_RETURN field descriptions

Field	Description
BITS	Reflects value of OTP Bank 5, word 6 (ADDR = 0x2e).

46.5.41 Value of OTP Bank5 Word7 (HW Capabilities) (OCOTP_SRK_REVOKE)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 5, word 7 (ADDR = 0x2f).

EXAMPLE

```
Empty Example.
```

Address: 21B_C000h base + 6F0h offset = 21B_C6F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_SRK_REVOKE field descriptions

Field	Description
BITS	Reflects value of OTP Bank 5, word 7 (ADDR = 0x2f).

Chapter 47
On-Chip RAM Memory Controller (OCRAM)

47.1 Overview

The on-chip RAM is implemented as an slave device on the 64-bit system AXI bus. Designed as a simple on-chip memory block, it has one bank of single port SRAM and supports one AXI port.

For the AXI port, the read and write transactions are handled by two independent blocks. As it is possible to have simultaneous read and write requests from the AXI bus, an arbiter with round-robin scheme is implemented to handle this. After arbitration, the granted read or write access command can then be issued to the memory cell through a read/write MUX.

Various options are provided for adding pipeline or wait-states in read/write access, in order to ensure flexible timing control at both high and low frequencies.

The internal block diagram is shown in the figure below.

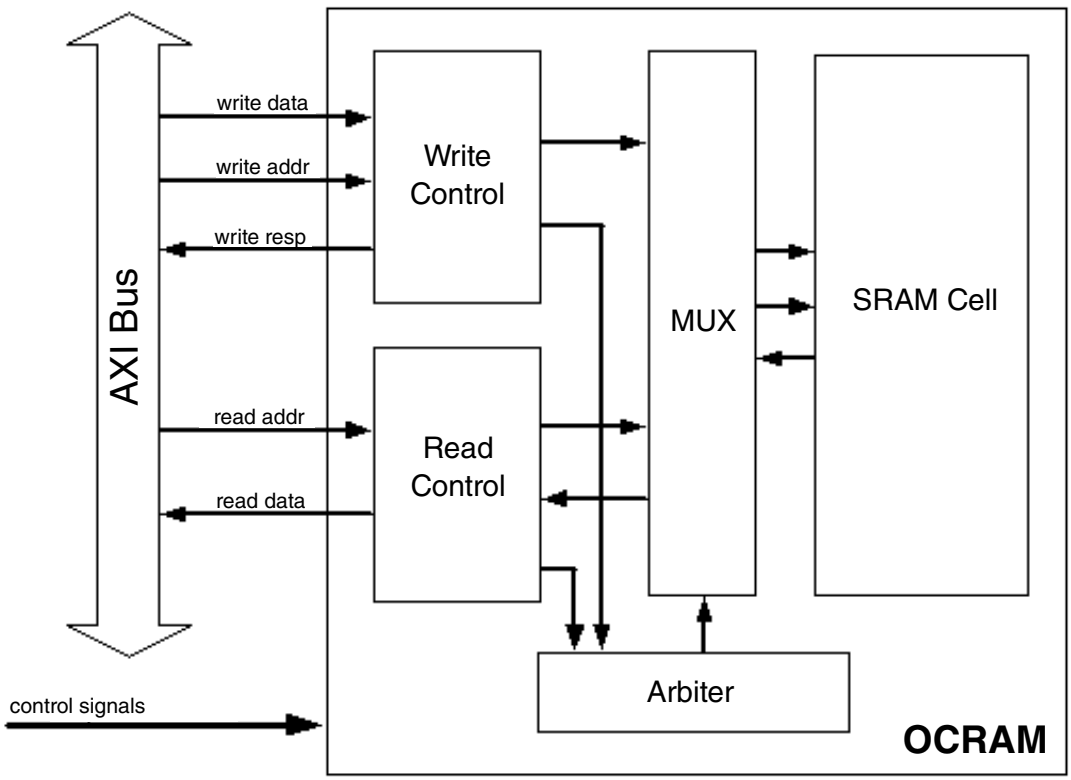


Figure 47-1. On-chip RAM Block Diagram

47.2 Basic Functions

47.2.1 Memory Map

The on-chip RAM size for i.MX 6Dual/6Quad is 256 Kbytes, organized as 16K x 64 bits, mapped from 0x00900000 to 0x0093FFFF.

Memory alias is also supported for on-chip RAM. Address from 0x00940000 to 0x009FFFFF are alias addresses for it, any read/write operation to this area will be mapped back to the 256 Kbytes area.

The 32-bit AXI address are arranged as shown in the table below.

Table 47-1. On-Chip RAM Address Bits

AHB ADDR Bits	Usage	Description
29:3	Address	Selects one of the 16 K words in the memory(64-bit a word).
2:0	Byte Address	Selects/masks out specific bytes within a word.

47.2.2 Read/Write Arbitration

The read/write arbitration uses the round-robin method.

The detailed rules used in arbitration are as follows:

- If there is no granted read or write in the last cycle, and there is only a read request or a write request, the request will be granted.
- If there is no granted read or write in the last cycle, and there are both read or write requests coming in at the same time, the read request will be granted first.
- If a granted read/write transaction has just finished, the write/read request will have the higher priority in the next cycle.
- If the first read/write access request in a transaction is granted, all the data transfer in this burst will be finished before the next arbitration begins, that is, the round-robin arbitration mechanism is based on AXI transaction, not data access.

47.2.3 TrustZone

TrustZone is also supported on this block.

When SECURE_ENBL bit in the General Purpose Register (IOMUXC_GPR10) bits [10:4] and [26:20] is set, the STARTADDR and ENDADDR bit-fields in this register establish the region of OCRAM that can only be accessed (both read and write) according to the execution mode policy described in CSU chapter, Peripheral access policy. If this bit is cleared to zero, the entire OCRAM can be accessed in either secure or non-secure mode. The TrustZone bits shows in Programmable Registers.

NOTE

The ENDADDR is not configurable and its value is the last address of the OCRAM space. The STARTADDR granularity is of 4KB.

47.3 Advanced Features

This section describes some advanced features designed to avoid timing issues when the on-chip RAM is working at high frequency.

All of the features can be disabled/enabled by programming the corresponding fields of the General Purpose Regsiter (IOMUXC.GPR3) bits [24:21] in the IOMUX chapter.

47.3.1 Read Data Wait State

When the wait state is enabled, it will cost 2 cycles for each read access, (each beat of a read burst).

This can avoid the potential timing problem caused by the relatively longer memory access time at higher frequency.

When this feature is disabled, it only costs 1 clock cycle to finish a read transaction, that is, to get read data back in the next cycle of read request becomes valid on the bus.

The read data wait state is configurable via IOMUXC.GPR3[21].

47.3.2 Read Address Pipeline

When this feature is enabled, the read address from the AXI master is delayed 1 cycle before it can be accepted by the on-chip RAM.

This can avoid setup time issues for the read access on the memory cell at high frequency. Enabling this feature can cost, at most, 1 more clock cycle for each AXI read transaction, that is, at most 1 more clock cycle for each read burst with multiple beats of data.

When this feature is disabled, the read address from the AXI master can be accepted by the on-chip RAM without delay, and data can become ready for master at next clock cycle (if no other access and no read data wait).

The read address pipeline is configurable via IOMUXC.GPR3[22].

47.3.3 Write Data Pipeline

When this feature is enabled, the write data from the AXI master would be delayed 1 cycle before it can be accepted by the on-chip RAM.

This can avoid setup time issue for the write access on the memory cell at high frequency. Enabling this feature would cost at most 1 more clock cycle for each AXI write transaction, that is, at most 1 more clock cycle for each write burst with multiple beats of data.

When this feature is disabled, the write data from the AXI master can be accepted by the on-chip RAM without delay, and data can be written to memory at this cycle (if no other access and write address is also ready at this cycle).

The write data pipeline is configurable via IOMUXC.GPR3[23].

47.3.4 Write Address Pipeline

When this feature is enabled, the write address from the AXI master would be delayed 1 cycle before it can be accepted by the on-chip RAM.

This can avoid setup time issue for the write access on the memory cell at high frequency. Enabling this feature would cost at most 1 more clock cycle for each AXI write transaction, that is, at most 1 more clock cycle for each write burst with multiple beats of data.

When this feature is disabled, the write address from the AXI master can be accepted by the on-chip RAM without delay, and data can be written to memory at this cycle (if no other access and write data is also ready at this cycle).

The write address pipeline is configurable via IOMUXC.GPR3[24].

47.4 Programmable Registers

There are no programmable registers in this block; however, OCRAM configurable bits can be found in the IOMUX Controller (IOMUXC) general purpose registers found here.

- TrustZone bits: IOMUXC_GPR10
- WAIT state / Pipeline bits: IOMUXC_GPR3
- L2 Cache OCRAM enable bits: IOMUXC_GPR11

(See IOMUXC Memory Map/Register Definition for more details).

Chapter 48
PCI Express (PCIe)

48.1 Overview

PCI Express includes the following cores:

- PCI Express Dual Mode (DM) core
- PCI Express Root Complex (RC) core
- PCI Express Endpoint (EP) core

NOTE

Throughout this chapter, content that is specific to a core is identified by using the abbreviated core name; for example, DM, RC, EP. Content that applies to the DM core specifically in either RC mode or EP mode is further qualified by using RC Mode or EP Mode. Content that applies to all cores is not identified as being specific to any core; rather, the term core or cores is used.

48.1.1 Terms and Abbreviations

The following terms are used throughout this chapter:

Table 48-1. Terms and Abbreviations

Term	Description
DM	PCI Express Dual Mode (DM) core
RC	PCI Express Root Complex (RC) core
EP	PCI Express Endpoint (EP) core
PCIe	PCI Express
CXPL	Common Xpress Port Logic: An internal Port Logic core module that implements the majority of the PCI Express protocol
x1/x2/x4/x8/x16	1/2/4/8/16 lanes

Table continues on the next page...

Table 48-1. Terms and Abbreviations (continued)

PIPE	PHY Interface for the PCI Express Architecture
NW	Number of double words; 1 stands for a 32-bit DWORD
DW	Data width: 32, 64, or 128 bits
NF	Number of functions; 1 stands for one function
TLP	Transaction Layer Packet
DLLP	Data Link Layer Packet
VC	Virtual Channel
BAR	Base Address Register
XADM	Transmit application-dependent module.
RADM	Receive application-dependent module.
PMC	Power management controller.
RAMI	External RAM interface.
Core	Identifies the entire core. The core includes the native PCIe-core and AXI/AHB bridge if present.
Native PCIe core	Identifies the basic PCIe core which has its own non-standard, proprietary dedicated bus interface to the application.
CDM	Configuration Dependent Module This is an internal block in the native core that houses the PCI configuration registers and some user-accessible registers that reside in the core. In general, an application may use our core, but will add other registers that are unique to their applications. Those new application registers will be referred to as External Application Registers (EAR).
LBC	Local Bus Controller This is an internal block that resides in the native core. It allows the DBI interface (from the application side) or the wire side interface (via the radm_TRGT0 interface) to access the core's Configuration Dependent Module (CDM) registers and/or the External Application Registers (EAR). (For additional details on this module, please refer to the native core documentation.)
ELBI	External Local Bus Interface This is an interface on the native core that processes read/write access to the external application registers (EAR). For applications that require external registers, the application can access the EAR through the bridge or an entirely different interface (outside the scope of this core). (For additional details on this interface, please refer to the native core's documentation.)
DBI	Data Bus Interface This bus is internal to the native core. This interface is used to access the core's internal registers (CDM) or the external application's device-specific registers attached to the PCIe native core ELBI interface.
VMI	Vendor Message Interface.
Inbound traffic	PCIe transactions that enter the native core from the wire side of the core (PCIe wire). These transactions will be delivered to the application side.
Outbound traffic	Transactions that enter the native core from the application side of the core. These are passed to the native core, where they will be sent out onto the PCIe wire.
MTU	Maximum Transfer Unit Specifies the maximum packet payload size supported. This indicates the maximum allowed transfer size for a write or completion.
Page boundary	Specifies the address page boundary size supported by the bridge. No packet can have an address that crosses the specified address boundary.
big-endian	Data format in which most significant byte comes first; normal order of bytes in a word.

Table continues on the next page...

Table 48-1. Terms and Abbreviations (continued)

RBYP	Receive Bypass Interface
RTRGT1	Receive Target 1 Interface
RCPL	Receive Completion Interface
XALI0/1/2	Transmit Application Client Interfaces
SII	System Information Interface.
iATU	Internal Address Translation Unit.

48.2 Architecture

This information describes the architecture of the PCI Express core.

The topics for this section are:

- Common Xpress Port Logic (CXPL)
- Transmit Application-Dependent Module (XADM)
- Receive Application-Dependent Module (RADM)
- Configuration-Dependent Module (CDM)
- Power Management Controller (PMC)
- Local Bus Controller (LBC) and Data bus Interface (DBI)
- Message Generation
- Debug and Diagnostics

The Common Xpress Port Logic (CXPL) module implements the basic functionality for the PCI Express Physical, Link, and Transaction Layers. In addition to the CXPL, there are several top-level modules that provide the configurationim and mode- specific features:

- Transmit Application-Dependent Module (XADM)
- Receive Application-Dependent Module (RADM)
- Configuration-Dependent Module (CDM)
- Power Management Controller (PMC)
- Local Bus Controller (LBC) and Data bus Interface (DBI)
- Message Generation

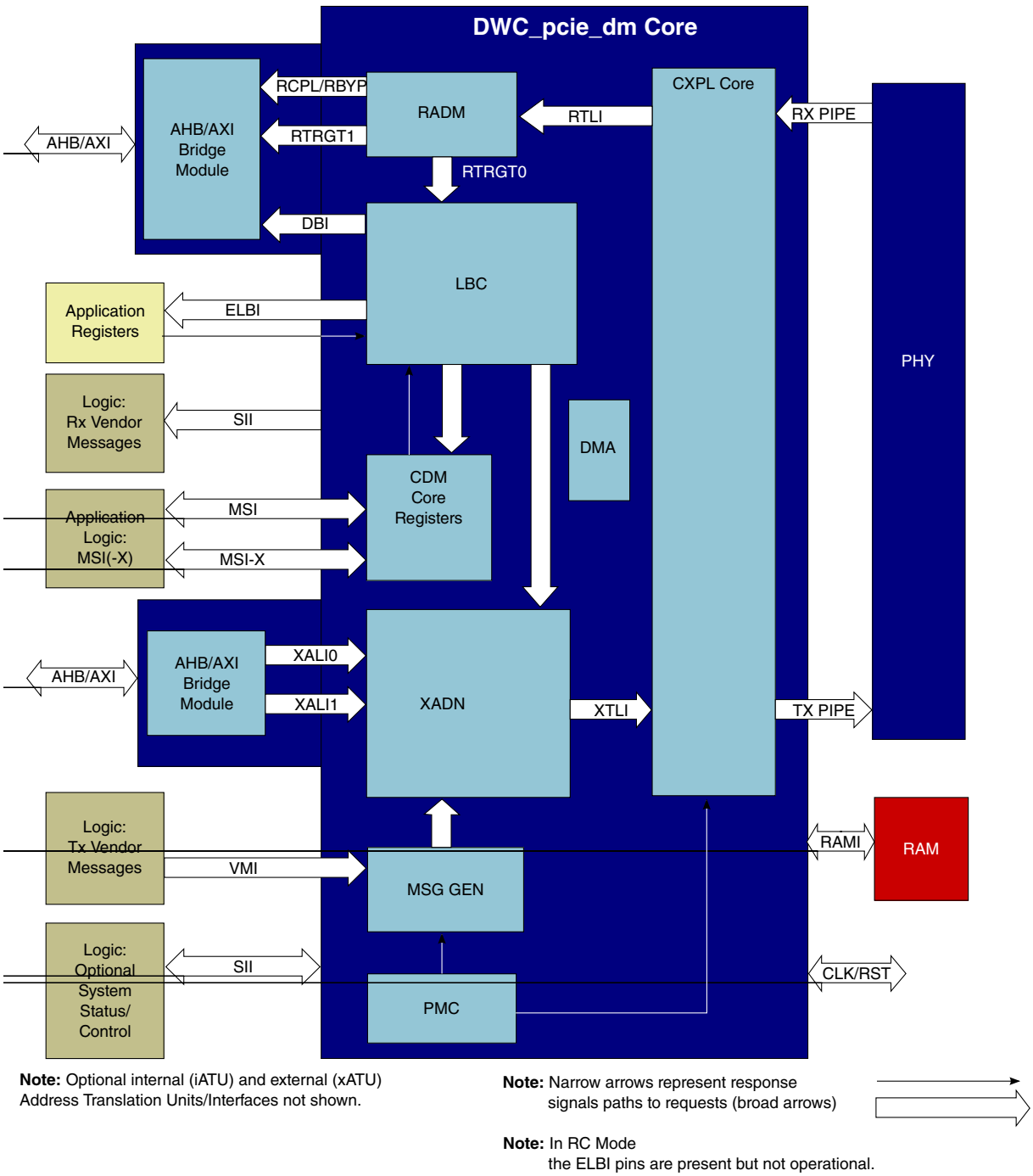


Figure 48-1. DM Core Block Diagram (with AXI Bridge Module)

NOTE

- Optional internal (iATU) and external (xATU) Address Translation Units/Interfaces not shown.
- Narrow arrows represent response signal paths to requests (broad arrows)
- In RC mode the ELBI pins are present but not operational.

For definitions of acronyms used for block and interface names, see Terms and Abbreviations.

Table 48-2. Core Interface Summary

Interface	Function
Transmit Client 0 Interface (XALI0)	Transmit interface for outbound Request or Completion TLPs.
Transmit Client 1 Interface (XALI1)	Additional application transmit interface, identical to XALI0. The usage of XALI0 and XALI1 is up to the application. For example, an application may use XALI0 for Requests and XALI1 for Completions.
Receive Completion Interface (RCPL)/Bypass Interface (RBYP)	When the core is configured in single or multiple queue architecture, RCPL delivers Completions received by the core to the application client that requested them. When the core is configured in segmented buffer queue mode, Bypass delivers all transactions that are configured as bypass.
Receive Target 1 Interface (RTRGT1)	Delivers inbound Requests received by the core after the Requests are qualified by the filter rules of the core.
Data Bus Interface (DBI)	Delivers an RD/WR request from application logic such as EEPROM to internal registers of the core or application registers at ELBI.
Message Signaled Interrupt (MSI) Interface	Allows the application to request a transmission of an MSI independent from the client interfaces. MSI interface is used only in EP mode.
MSI-X Interface	Allows the application to request a transmission of an MSI-X independent from the client interfaces. MSI-X interface is used only in EP mode.
Vendor Message Interface (VMI)	Allows the application to request a transmission of a vendor message independent from the client interfaces.
System Information Interface (SII)	Exchanges system information between the core and the application.
PIPE	Standard PIPE interface between the PCI Express PHY and the core.
RAM Interface (RAMI)	Optional top-level interface to connect external RAMs for the retry buffer and receive queues. If you do not select the optional top-level RAMI, the RAMs reside inside the top-level hierarchy of the core.

48.2.1 Common Xpress Port Logic (CXPL)

The CXPL module implements a large portion of the Transaction Layer logic, all of the Data Link Layer logic, and the MAC portion of the Physical Layer, including the Link Training and Status State Machine (LTSSM).

The CXPL connects to the external PHY though the PIPE.

Important aspects of the CXPL and overall core implementation include:

- Layer 3 (Transaction Layer) functionality is split between the XADM, RADM, CDM, and CXPL.

- Layer 1(Physical Layer) is split across the PIPE such that the MAC functionality is in the core and the PHY functionality is implemented in the PIPE-compliant PHY.the PHYmodule resides outside of the core, interfacing through the standard PIPE.
- Receive and transmit path functionality is decoupled except where communication between the two is required (such as Flow Control and other low-level Link management functions).

CXPL contains six modules, three for transmission and three for reception, as shown in the figure below.

- RTLH: Receive Transaction Layer Handler
- XTLH: Transmit Transaction Layer Handler
- RDLH: Receive Data Link Layer Handler
- XDLH: Transmit Data Link Layer Handler
- RMLH: Receive MAC Layer Handler
- XMLH: Transmit MAC Layer Handler

CXPL is compliant with the PCI Express 3.0 Specification with regards to the physical layer, data link layer and transaction layer.

1. PCS soft logic and timing model for mixed signal PMA

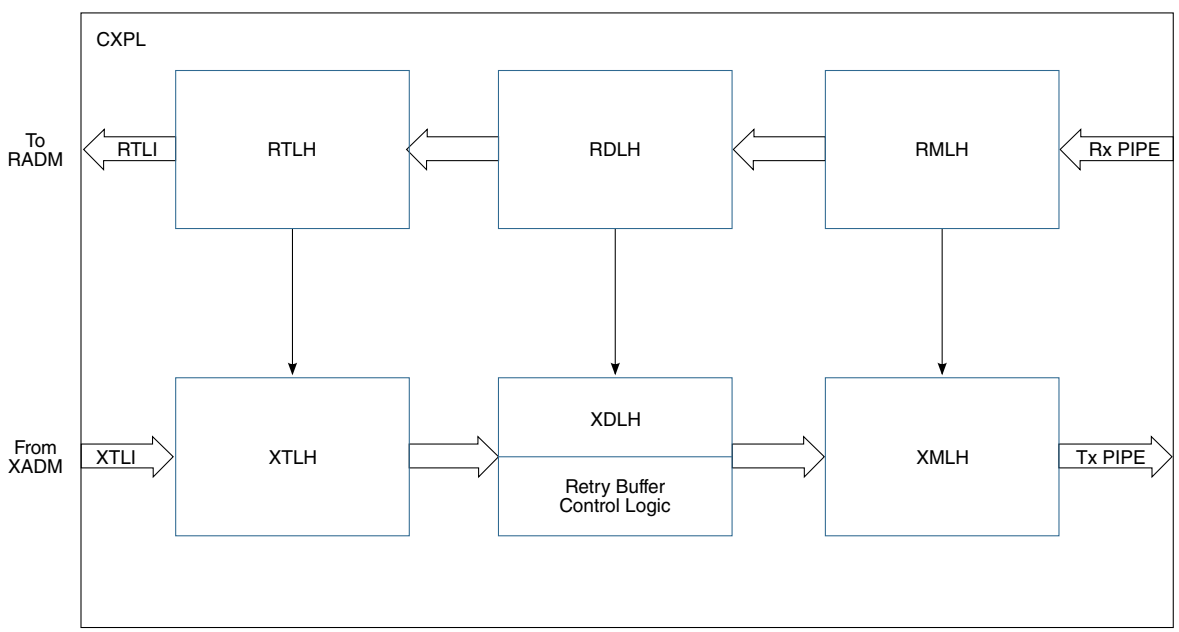


Figure 48-2. CXPL Module Block Diagram

48.2.2 Transmit Application-Dependent Module (XADM)

The XADM sits between the application logic and the CXPL core and implements the mode-specific functionality of the PCI Express Transaction Layer for packet transmission.

The figure below is a block diagram of the XADM. Its functions include arbitration, TLP formation, and credit checking.

The transmit path uses a cut-through architecture. It does not implement transmit buffering/queues (other than the retry buffer). Depending on system design, an externally-implemented transmit queue can be used to handle rate matching if the CXPL and application transfer rates are different. For relevant information on this in the context of using the AHB/ AXI Bridge as an application to the XADM.

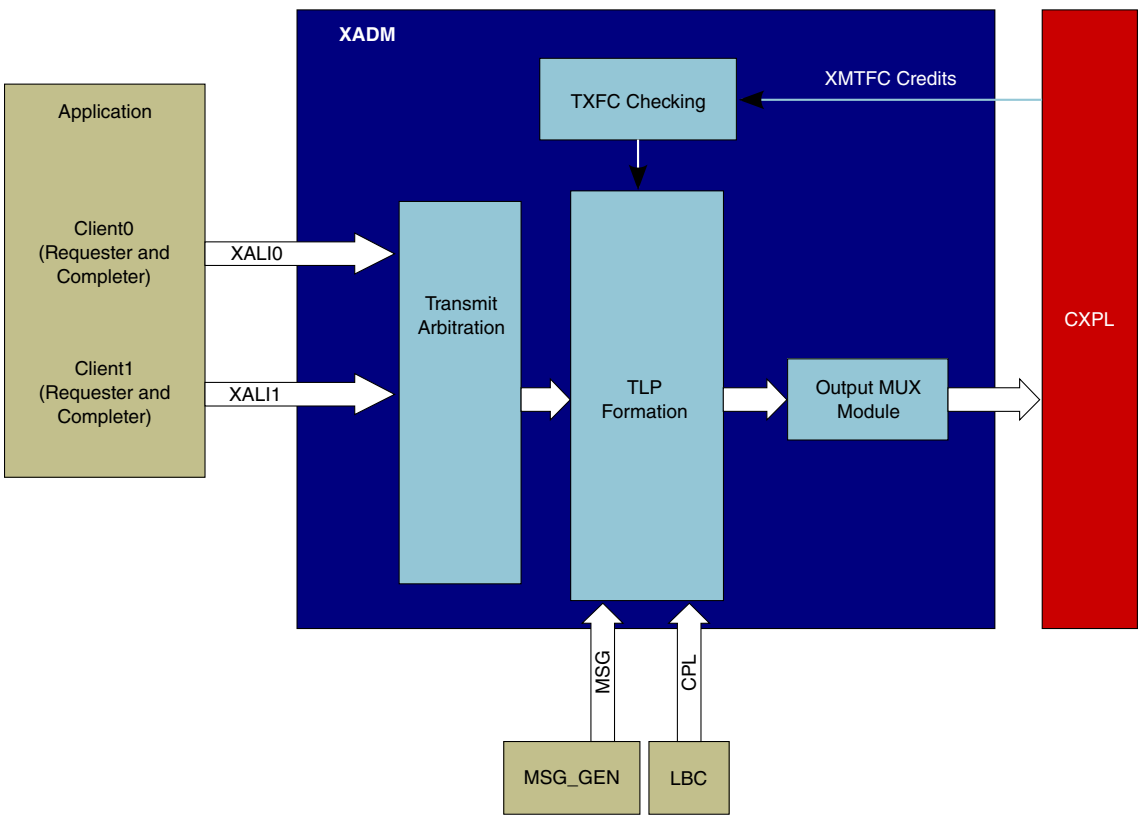


Figure 48-3. XADM Block Diagram

48.2.2.1 Arbitration

XADM provides the arbitration of TLP transmission between the following:

- The transmit client interfaces (XALI0, XALI1)
- Internally generated Messages from the MSG_GEN, triggered by PME, INTx (EP mode), errors, or application logic
- Internally generated Completions:
- EP mode: Internally generated Completions are responses for type 0 Configuration Read and Write Requests from upstream components, memory or I/O-mapped application register space Read and Write Requests, or responses to error conditions (Unsupported Requests).
- RC mode: Internally generated Completions are Unsupported Request or Completer Abort, as required by the incoming Request filtering function of the RADM.

In general, all internally generated TLP requests have higher priority than client interfaces.

For details about how the arbitration methods work and how to configure the arbitration methods, see Transmit TLP Arbitration.

48.2.2.2 Credit Checking

The core checks that enough FC credits are available in the remote device for the specific type of transaction (P, NP, CPL) before allowing a transmission of a TLP. TLPs that passed the credit check are arbitrated according to the supported arbitration method. Internally generated Completions and Messages are also gated by the arbitration logic, though at highest priority, and must also pass the FC credit test before they are accepted for transmission.

48.2.3 Receive Application-Dependent Module (RADM)

The RADM sits between the application logic and the CXPL core and implements the mode-specific functionality of the PCI Express Transaction Layer for TLP packet reception.

The figure below shows a block diagram of the RADM. The RADM serves four major functionalities as following:

- Sort/Filter received TLPs
- Completion Lookup Table (CLT), which is used for Completion tracking and Completion timeout monitoring of transmitted Non-Posted requests.
- Provide queuing (or bypass) of the received TLP
- Output received TLP to the core's receive interface (Demux function)

The filtering rules and routing for all TLP receive options are configurable for all TLPs received.

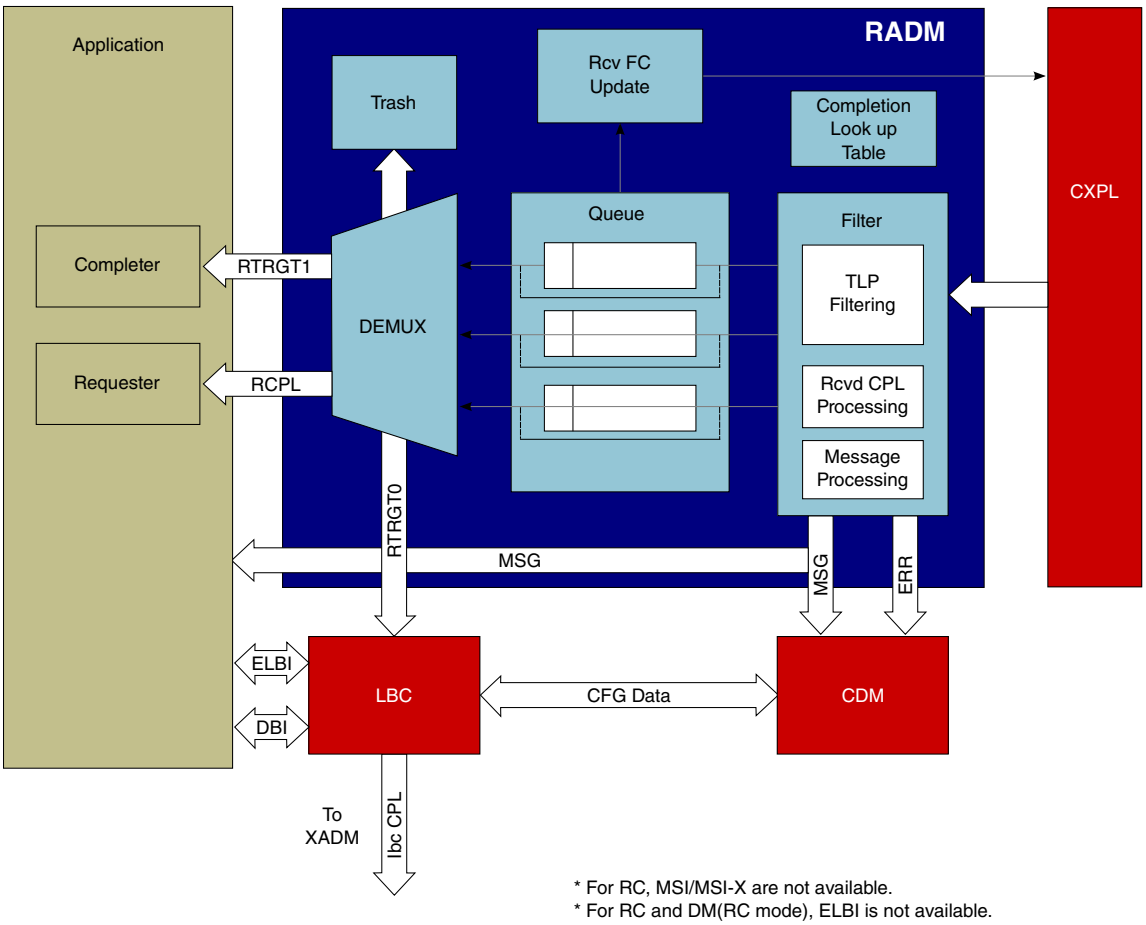


Figure 48-4. RADM Block Diagram

48.2.3.1 Posted and Non-Posted Request and Completion TLP Processing

The RADM filter passes the Posted and Non-Posted Request and Completion transactions (such as Write Transactions and Memory Reads) directly to the application through the RTRGT1 interface or to RTRGT0 for internal modules, as determined by the filtering and routing rules for the current operating mode, as described in Receive Filtering.

The RADM filter segregates Posted and Non-Posted TLPs into valid supported and valid un-supported Requests, and forwards them to the queue. The filter processes each Request and determines each TLP's destination along with other controls that may be needed to generate TLPs.

For Requests that the core forwards to the RTRGT1 or Bypass interface, the application must process the Request and generate the Completion.

For Requests that the core forwards to RTRGT0, the core automatically generates the Completion. The core automatically executes any required ELBI access before generating the Completion.

48.2.3.1.1 TLP Routing

The RADM demux is designed to mux out a received TLP to the RTRGT1 and RCPL/RBYP interfaces from single queue or multiple queue (DM/RC/EP) configurations. The filter determines the destination and the action for each TLP, then sends this to the queue. The demux decides whether to discard or forward the TLP onto the RTRGT1, RTRGT0, RCPL or RBYP interfaces.

For more details see Receive Routing and Queue to Port Mapping.

48.2.3.2 Received Completion TLP Processing

Received Completions are filtered against the completion lookup table content before presenting the Completion to the queue.

The RADM also implements a Completion time-out mechanism (via the Completion Lookup Table) and notifies the application when an expected Completion-corresponding to a transmitted Non Posted TLP-does not arrive within a specified time.

The Completion Lookup Table (and Completion Timeout event) should not be confused with the Target Completion Lookup table (and Target Completion Timeout event) .

The Target Completion Lookup Table is watching for received application completions (on XALI0/1) corresponding to previously received Non Posted requests.

The Completion Lookup Table is watching for received PCIe completions corresponding to previously transmitted Non Posted requests.

Typically, infinite Completion credits are advertised and the received completion is configured in bypass mode which means that there is no queue in the core to store completions.

NOTE

It is fully expected that the application will have enough buffering space ready for its requests, so no backpressure mechanism is needed. By default, the Completion queue operates in bypass mode.

Completions can be configured in store and forward mode if the application has chosen to do so. If a completion lookup has failed or other completion filtering has failed, the core will assert an abort signal at the end of the transaction. If the core is configured to have Completions in bypass mode, it is the application's responsibility to roll back any actions at the application's queue when an abort signal is asserted. If the core is configured with Completions enqueued, the Completion will be discarded by the core and flow control credits will be updated, as necessary, when an abort signal is detected.

48.2.3.3 Message Processing

The RADM filter provides a Message interface (grouped as part of the SII) to handle the Message TLPs received from the upstream component. The RADM filter processes the Message and decodes the header before sending it to the application logic on the SII. You can also select a configuration option to send the entire Message TLP to the application in addition to providing the decoded Message on the SII. For more details see Message Reception.

48.2.4 Configuration-Dependent Module (CDM)

The CDM implements the standard PCI Express configuration space and the core-specific register space. The CDM also requests the Message generation module to send Messages, as required, including MSI and interrupts.

The specific PCI Express configuration structures implemented in the CDM include the following:

PCI-Compatible Configuration Registers

- RC mode: Type 1 header
- EP mode: Type 0 header

PCI Capability Structures:

- PCI Power Management Capability Structure
- MSI Capability Structure

- MSI-X Capability Structure
- VPD (Vital Product Data) Capability

PCI Express Capability Structure

PCI Express Extended Capabilities:

- Advanced Error Reporting Capability
- Virtual Channel Capability
- Device Serial Number Capability
- Power Budgeting Extended Capability

The configured device type (determined by the device_type[3:0] input signal) affects the behavior of the Message generation engine, error reporting mechanism, as well as some PCI Express configuration space registers.

The CDM communicates with application's host bus controller through the DBI. The host bus controller controls accesses to registers within each CDM in multiple instances of the core in a multi-port design.

48.2.5 Local Bus Controller (LBC) and Data bus Interface (DBI)

This following topics are covered in this section:

- Overview (LBC)
- ELBI
- CDM Register Space Layout
- PCI Configuration Header and Capability Registers (in CDM)
- Port Logic (PL) Registers (in CDM)
- PCIe Wire Access (EP mode)
- PCIe Wire Access (RC mode)
- DBI Access
- LBC/DBI Feature Availability
- LBC/DBI Size Limitations
- AXI DBI Limitations

48.2.5.1 Overview (LBC)

The LBC module provides a mechanism for a link partner PCIe device (in EP mode only) or a local CPU (through the DBI) to access internal registers (in the CDM) .

NOTE

In RC mode:

The application can access CDM registers through the DBI.

PCIe wire access (through RTRGT0) to the CDM registers is NOT possible.

Figure 48-5 shows the location of the LBC within the PCIe core and its role in routing transactions.

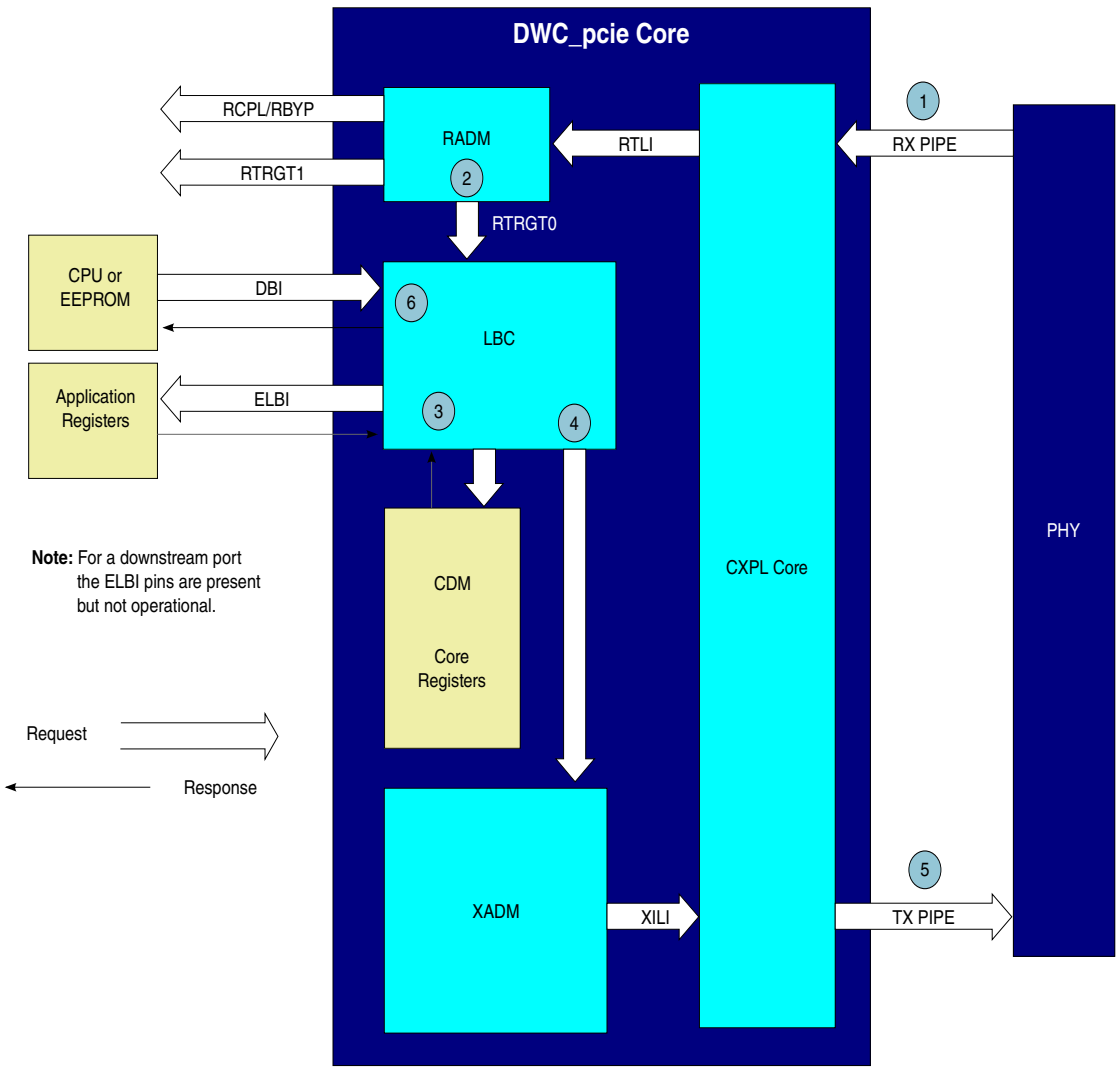


Figure 48-5. LBC Context

The LBC provides a Switched access function to internal registers (in the CDM)from the local application processor (CPU) via the DBI or the remote application software (off the PCIe RX wire) via RTRGT0. Figure 48-6 illustrates the four possible request paths through the LBC.

For more information on the filtering and routing of received inbound TLPs, see Receive Routing.

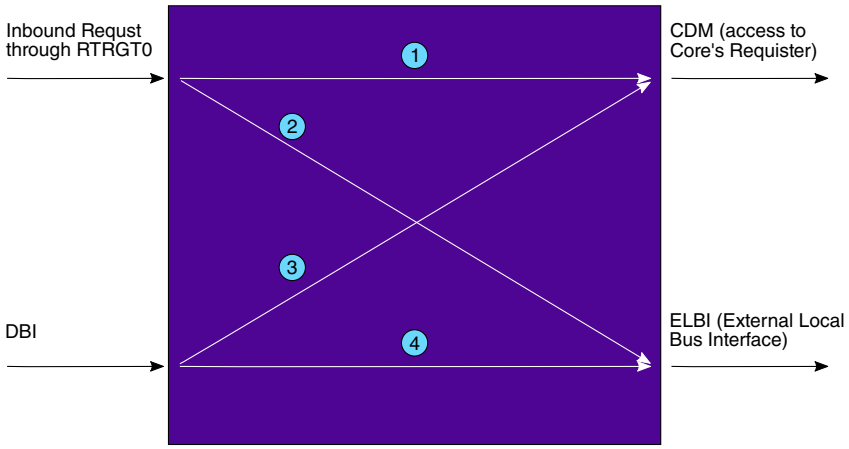


Figure 48-6. LBC Switch

NOTE

In RC mode PCIe wire access (through RTRGT0) to the CDM registers is NOT possible.

48.2.5.1.1 Simultaneous Transactions

The LBC is single-threaded and therefore, the DBI and RTRGT0 cannot use the LBC at the same time. For example, a request on the DBI will not be accepted, during a RTRGT0 transaction, until both parts of that transaction - [1] request and [2] response (completion generation) - are completed.

If the DBI and RTRGT0 present a request at the same time (regardless of the target/destination of each request), then the LBC will grant access to the RTRGT0.

48.2.5.2 ELBI

You can connect external application registers to the ELBI. These can be accessed by PCIe request TLPs over the PCIe link or by the DBI.

NOTE

In RC mode PCIe wire access (through RTRGT0) to the ELBI is NOT possible.

48.2.5.3 CDM Register Space Layout

The core has 4096 bytes of PCI Express configuration space per function distributed as per the figure below. This address space is fully accessible from the DBI without any restrictions.

In EP mode it can be accessed from the PCIe wire using CFG requests. Under certain configurations, the Port Logic and ELBI Register spaces can also be accessed from the PCIe wire with MEM and IO requests.

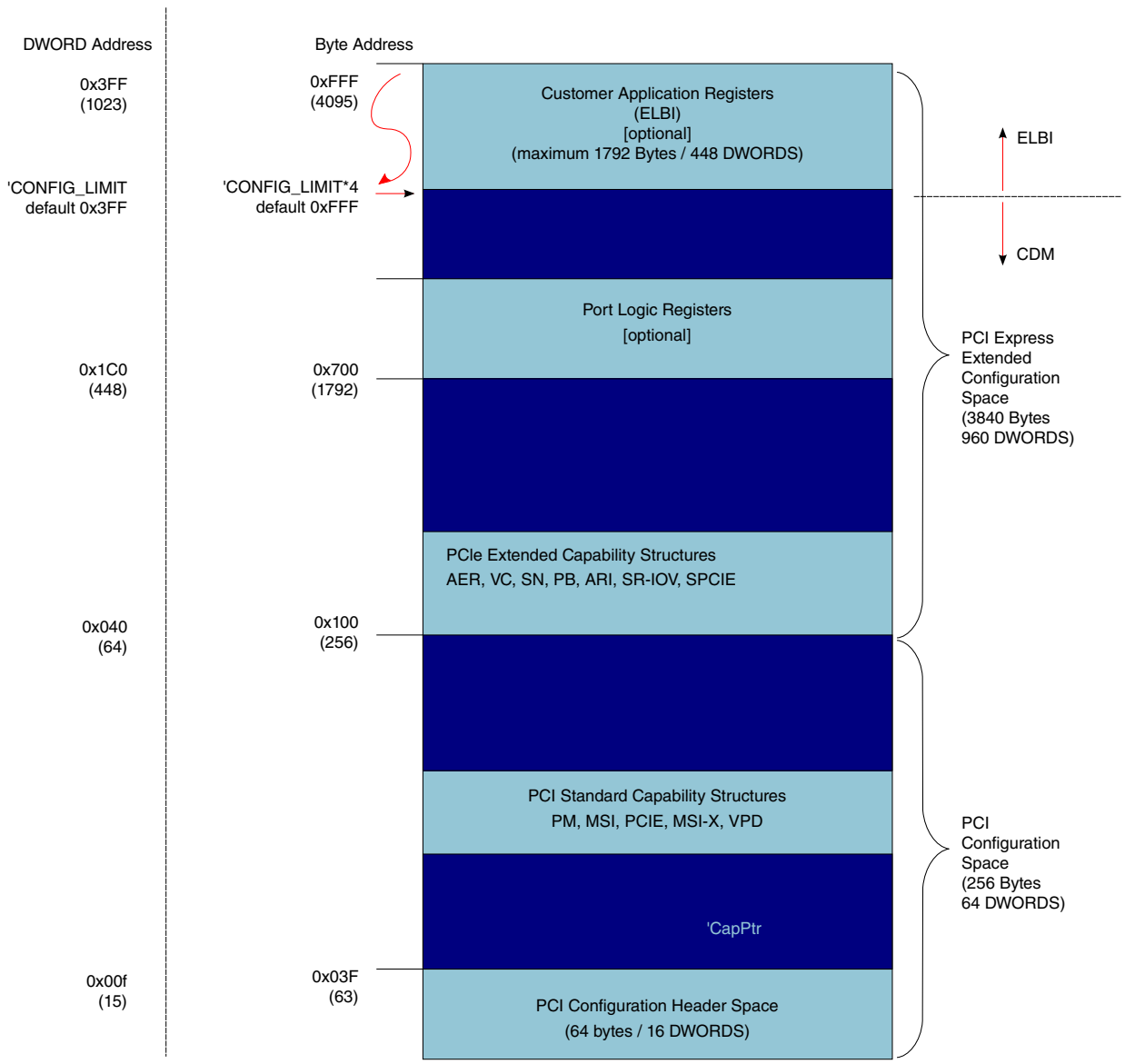


Figure 48-7. PCIe Core Configuration Space Address Map (per function)

A CFG TLP has a 6-bit Register Number Field and a 4-bit Extended Register Number field allowing 1024 DWORDS (4096 bytes) to be accessed.

48.2.5.3.1 PCI Configuration Header and Capability Registers (in CDM)

The PCI Configuration Header and Capability Registers in Figure 48-7 are PCIe core configuration registers specified by the PCI Express 3.0 Specification. Access from the PCIe wire is possible with CFG requests (in EP mode only).

These registers are fully accessible from the DBI without any restrictions.

NOTE

From the PCIe wire (through RTRGT0) in EP mode only:

You can Memory-Map the Port Logic (PL) Register Space.

You cannot Memory-Map the PCI and PCIe Configuration Register Spaces. You must always access them with a CFG request.

NOTE

In RC mode:

PCIe wire access (through RTRGT0) to the CDM registers is NOT possible.

NOTE

From the DBI:

You can access without any restriction the Port Logic (PL) Register Space.

You can access without any restriction the PCI and PCIe Configuration Register Spaces.

48.2.5.3.2 Port Logic (PL) Registers (in CDM)

The Port Logic Registers in Figure 48-7 are PCIe core configuration registers not specified by the PCI Express 3.0 Specification, but are specific to the configuration and operation of the PCIe IP core.

In EP mode, access from the PCIe wire is with CFG requests. There is no access from the PCIe wire in RC mode.These registers are fully accessible from the DBI without any restrictions.

48.2.5.3.3 Memory Mapping PL Registers

Port Logic Registers (which by default are accessed by CFG requests) can also (at the same time) be accessed by MEM requests through the use of the ENABLE_MEM_MAP_PL_REG, PL_FUNC_NUM and PL_BAR_NUM configuration parameters. These can be used to map the Port Logic Registers to any BAR of any function.

All MEM requests that match PL_BAR_NUM (when ENABLE_MEM_MAP_PL_REG=1) and whose address offset is in the range 0x700-0x8FF will be routed to the Port Logic Registers.

NOTE

The BAR corresponding to PL_BAR_NUM must also be mapped/assigned to RTRGT0.

48.2.5.4 PCIe Wire Access (EP mode)

By default[1], CFG requests are routed to RTRGT0 and then to CDM via LBC.

By default, BAR-matched MEM/IO requests are routed to RTRGT1.

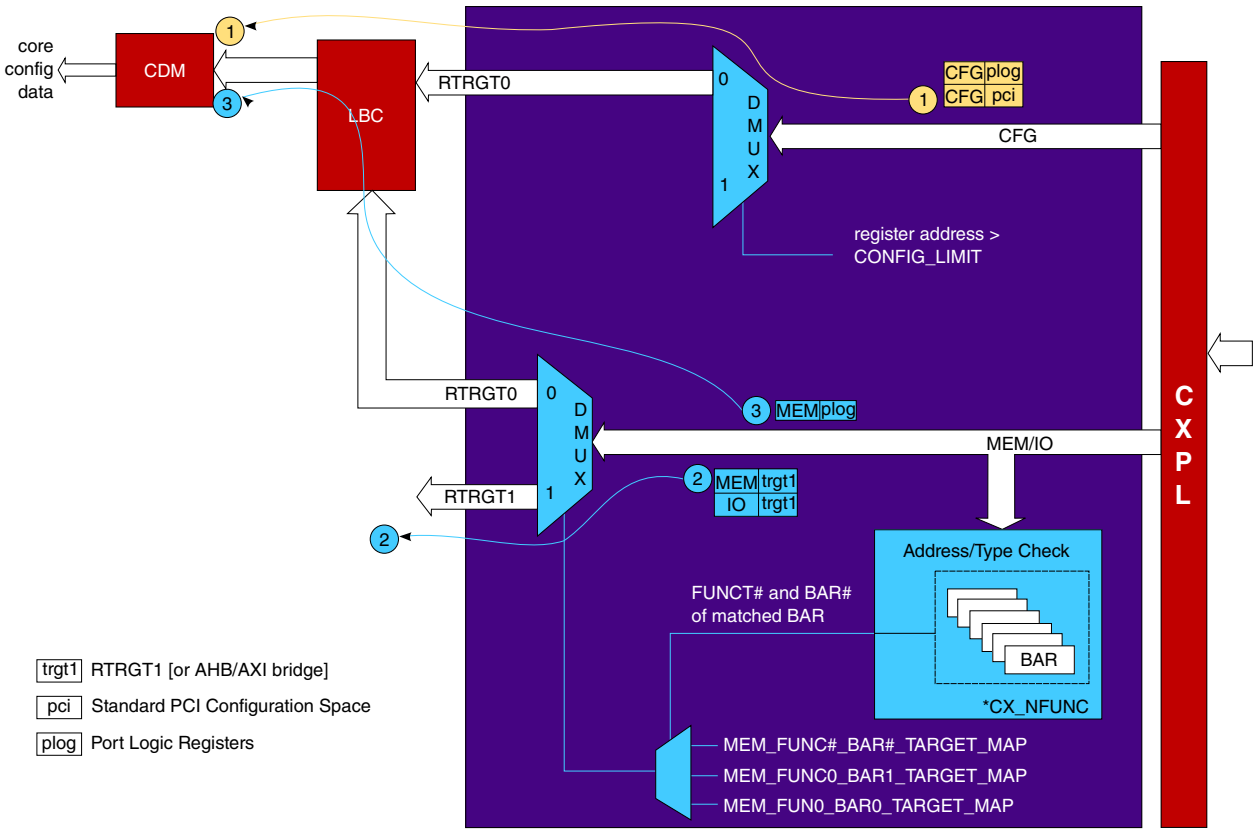


Figure 48-8. Request TLP Routing - Typical Use Model

Looking in detail at the typical routing of inbound PCIe requests (the numbers refer to the paths identified by the circled numbers in the above diagram.

CFG requests-either to Standard PCI Configuration Space or Port Logic Registers-are always routed to the CDM. This is because CONFIG_LIMIT by default is set to 0x3FF (top of the CFG register address space).

BAR-matched MEM/IO requests are routed to RTRGT1. MEM_FUNC#_BAR#_TARGET_MAP for each enabled BAR is set by default to RTRGT1.

Port Logic Registers (which by default are accessed by CFG requests-see [1] above) can also be accessed by MEM requests through the use of the ENABLE_MEM_MAP_PL_REG, PL_FUNC_NUM and PL_BAR_NUM configuration parameters. These can be used to map the Port Logic Registers to any BAR.

To see all routing possibilities, refer to Receive Routing.

48.2.5.5 PCIe Wire Access (RC mode)

To see all routing possibilities, see Receive Routing.

NOTE

In RC mode the application can access CDM registers through the DBI.

PCIe wire access (through RTRGT0) to the CDM registers is NOT possible.

48.3 Core Operations

This section describes the operations of the PCI Express core.

48.3.1 Initialization

Immediately after reset the DM core goes into either EP mode or RC mode depending on the state of the device_type input.

The internal configuration registers in the CDM assume their default reset values as listed in the following sections:

- PCIe Registers (EP mode)
- PCIe Registers (RC mode)
- PCIe Registers: Port Logic

The application must keep the app_ltssm_enable signal deasserted after reset until the application is ready to establish a Link and start receiving and transmitting TLPs. If the application needs to update configuration registers in the CDM as part of the initialization process, then the application must keep app_ltssm_enable deasserted until it has programmed all the necessary configuration registers through the DBI.

After initializing the necessary configuration registers, the application can assert app_ltssm_enable to allow the LTSSM to begin Link establishment. The LTSSM begins Link negotiation after the core_rst_n and phy_mac_phystatus signals are deasserted and app_ltssm_enable is asserted.

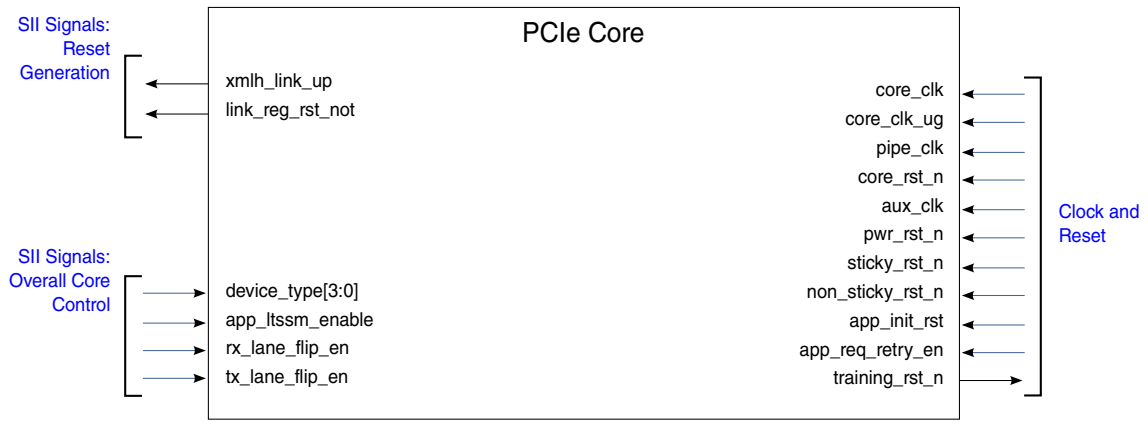


Figure 48-9. I/O Interfaces involved in Initialization

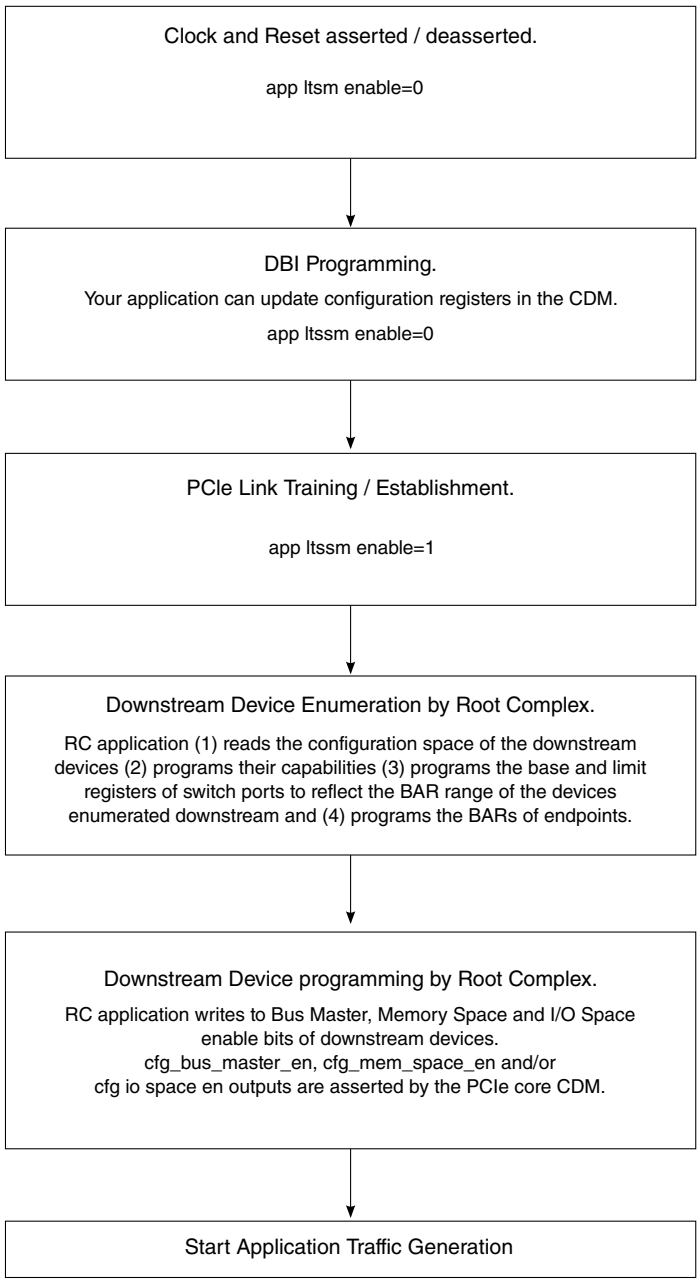


Figure 48-10. PCI Express Initialization Steps

48.3.2 Link Establishment

The core and a PCI Express compliant PHY combine to provide a complete solution for setting up and maintaining a compliant PCI Express Link.

The core implements the LTSSM function according to the *PCI Express 3.0 specification*.

In general, the process for establishing a Link is a follows:

1. Upon power-up (or directly out of reset), it is assumed that the power supply becomes stable and the PLLs reach frequency lock before the devices attempt to establish a valid Link. Once in a valid state, the SerDes either communicates a ready status to the core or simply begins transmitting and receiving valid data.
2. Per the *PCI Express 3.0 specification*, once bit and symbol synchronization are complete, the core initiates the following sequence to establish a Link (assuming a valid and properly functioning Link partner):

Receiver detection on available Lanes for the Port.

Exchange of Training Sequences to determine Link configuration (for example, Link speed, number of Lanes, and order).

Once both partners reach a valid negotiated state, the Link state is set up and the LTSSM is in L0.

1. Once Link up is achieved, the data link modules take over to manage the Link and initialize Flow Control.
2. After Flow Control initialization is complete, the data link modules signal the transaction layer modules that the link is ready to allow transmission/reception of TLP traffic.
3. During normal operation, the LTSSM and data link modules continue to manage the underlying Link integrity while data traffic is communicated across the PCI Express Link.

NOTE

The power management implementation also affects Link establishment.

48.3.3 Transmit TLP Processing

Information found here describes the flow of transmit TLPs through the core.

The topics for this section are:

- Transmit Overview
- Transmit TLP Arbitration
- Transmit Retry
- Transmit DLLP Priorities

It may helpful to first review the following sections: Common Xpress Port Logic (CXPL) and Transmit Application-Dependent Module (XADM).

48.3.3.1 Transmit Overview

Generally, all types of transmit TLPs (Posted, Non-Posted, and Completion) generated by the application travel through the core in the following flow:

1. The application presents a transaction transmission request with header information and payload (if applicable) on one of the transmit client interfaces (for example, XALI0).
2. The XADM forms the transaction into a TLP and checks the TLP against the current Flow Control credit availability. If the TLP passes the Flow Control checks and wins the arbitration with TLPs from the other the client interfaces, then the TLP goes to the CXPL.
3. The XTLH module inserts an ECRC (if applicable) and snoops/stores the necessary TLP information for Completion lookup (for Non-Posted requests only).
4. The XDLH inserts the Sequence Number and LCRC into the TLP and the retry buffer stores the TLP.
5. The XMLH inserts start and end delimiters and performs data scrambling.
6. The XMLH presents the packet to the PHY through the PIPE interface.
7. The PHY receives the packet, performs 8b10b encoding, and serialization, then sends the packet for transmission on the Link

The core does not check for TLP errors; instead it sends the TLP as presented on the XALI interface.

The native PCIe core does not check that the TLP payload size is less than the 'Maximum Payload Size' limit. However, the AXI bridge module does guarantee that this limit is not exceeded.

The native PCIe core does not check that the TLP payload size is less than CX_MAX_MTU limit. Exceeding this limit will overflow the retry buffer, resulting in data corruption. However, the AXI/AHB bridge module does guarantee that this limit is not exceeded.

48.3.3.2 Transmit TLP Arbitration

The transmit arbitration mechanisms supported by the core are as follows:

- Client-based round robin arbitration
- Strict priority client-based arbitration
- VC-based priority arbitration

To configure the transmit arbitration algorithm, use the 'Transmit Arbitration Method' configuration parameter (`CX_XADM_ARB_MODE).

Regardless of the transmit arbitration method selected, Messages (both internally-generated and Messages requested through the VMI) always have the highest priority, followed by internally-generated Completions. The priority order for all transmitted TLPs is:

1. Internally generated Messages
2. Internally-generated Completions
3. Transmit TLPs from Client0, Client1, and Client2 according to the selected arbitration method

48.3.3.2.1 Client-Based Arbitration

When you configure the core to use client-based arbitration (`CX_XADM_ARB_MODE = 1), the XADM uses round-robin arbitration between the two transmit client interfaces.

48.3.3.3 Transmit Retry

There is a Retry Buffer (RB) in the core that stores a copy of each transmitted TLP until an Ack is received. The RB consists of two buffers: retry buffer and start-of-TLP (SOT) buffer.

NOTE

The Retry Buffer does not function as a transmit queue. The core transmits TLPs immediately after they pass arbitration. The copy in the Retry Buffer is only sent in the event that the TLP must be re- transmitted.

The retry buffer is implemented with a single port RAM. The depth of the retry buffer is selected during hardware configuration either by enabling automatic buffer sizing or by setting the depth explicitly. The retry buffer width is set automatically (data bus width plus extra control bits).

The selected retry buffer size determines the size of the SOT buffer. The SOT buffer stores the starting address of each unacknowledged TLP stored in the retry buffer. The SOT buffer is implemented with a single port RAM and is indexed by the Sequence Number of the TLP whose starting address is being stored or retrieved.

The minimum depth of the SOT buffer is also a user configuration option. The selected size must allow the retry buffer to store the maximum number of shortest TLPs (3 DWORDs).

When a Nak is received or the replay timer times out, a replay is initiated. A replay is terminated by two conditions:

- When the replay of all TLPs in the retry buffer is finished, or
- An Ack DLLP is received that acknowledges all TLPs in the retry buffer

The replay timer tracks the TLP replay time. It stays at 0 when every TLP has received an Ack and starts to count when a TLP is transmitted and the LTSSM is not in the training state. The replay timer is reset to 0 when an Ack or Nak is received that acknowledges a TLP that is in the retry buffer.

48.3.3.4 Transmit DLLP Priorities

The order of priority to transmit pending DLLPs is:

1. High-priority DLLPs
2. TLPs
3. Low-priority DLLPs

48.3.4 Receive TLP Processing

The information found here describes the flow of receive TLPs through the core.

48.3.4.1 Receive Overview

It may helpful to first review the following sections:Common Xpress Port Logic (CXPL) and Receive Application-Dependent Module (RADM).

Generally, received transactions travel through the core in the following flow:

1. The PHY receives a stream of bits and aligns/forms them into 10-bit symbols (Gen1/2)
2. The PHY decodes the 10b stream into an 8b stream (Gen1/2)
3. The PHY crosses the clock domain from RX to TX and presents the stream to the PIPE.
4. The RMLH descrambles and deskews the incoming data, checks for receiver (Gen1/2) then extracts packets.
5. The RDLH strips off the LCRC and Sequence Number.
6. The RTLH strips off the ECRC (if applicable), checks for a malformed TLP, and forms a transaction across the RTLI interface to the RADM.

7. The RADM filters the transaction based on the transaction type (Posted, Non-Posted, or Completion) and the rules described in Receive Filtering.
8. Filtered transactions are sent to RADM queues.
9. Transactions residing in the RADM queues are presented to the application or locally handled by the LBC module, depending upon the filter result

48.3.4.2 Receive Filtering

The core contains a filter module that is responsible for the tasks listed here.

- Determine the status of a received TLP using filtering rules.
- Determine the destination interfaces of a received TLP based on the status from applying the filter rules.
- Signal the application for the status of the received TLP by driving signals such as DLLP abort, TLP abort and ECRC error.

The core filters and routes received TLPs according to a set of rules determined by the TLP type based on the *PCI Express Base 3.0 Specification* and user-configurable filtering options. The filtering rules for a received TLP are affected by I/O signals (run-time options), and register values (run-time options).

The application can mask some of the filtering and error handling rules by setting the corresponding bits in Symbol Timer Register and Filter Mask Register 1 and Filter Mask Register 2.

There are three types of the filtering rules in the core:

- 3.4.2.1: rules that are applicable for all TLP received
- 3.4.2.2: rules that are dependent on the type of the TLP based on PCIe specification
- 3.4.2.3: rules that are not from the PCIe specification but requested by specific applications.

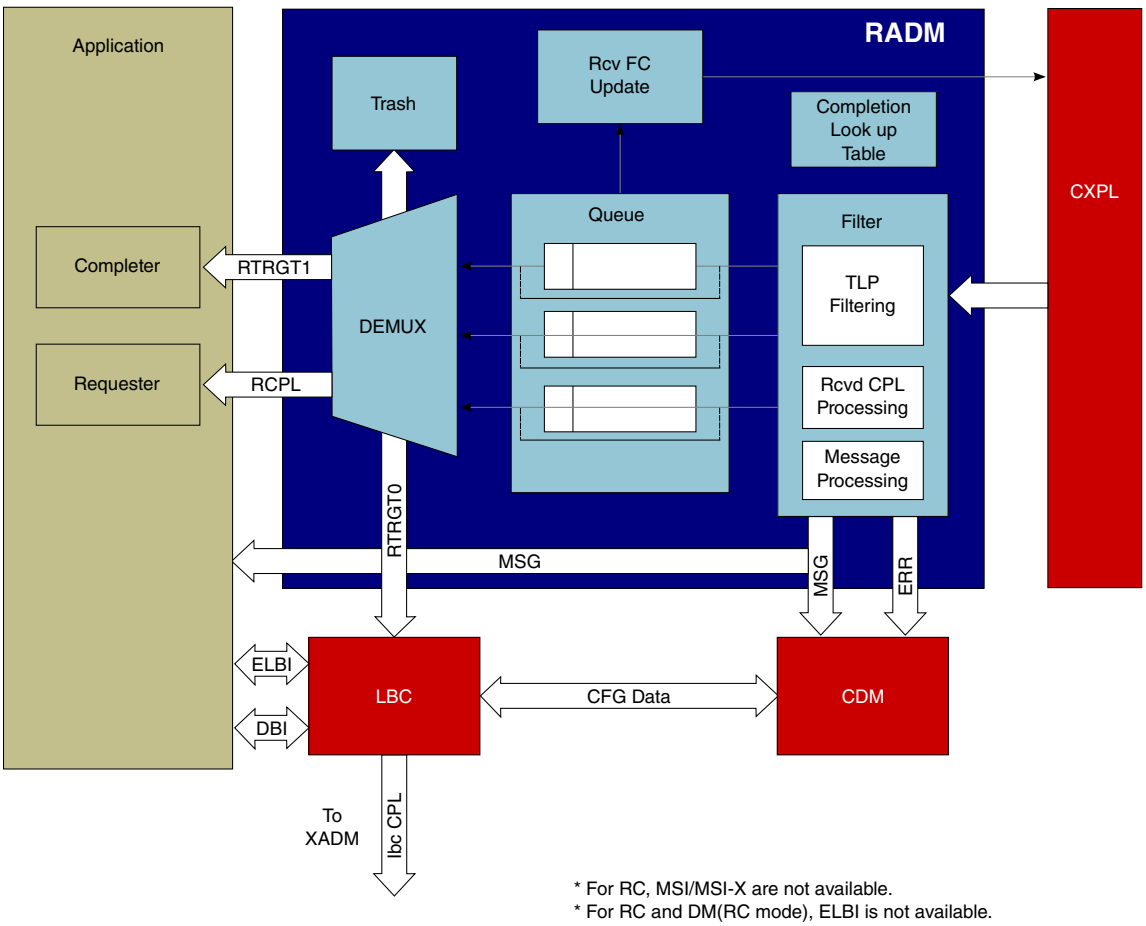


Figure 48-11. RADM Block Diagram

48.3.4.2.1 Filtering Rules Applicable for all TLPs Received

The general rules listed here apply to all incoming TLPs.

- The core discards all incoming TLPs that have an invalid Type field. This TLP is treated as a 'TLP ABORT'.
- By default, a request TLP with the poison bit set (EP=1) is considered an Unsupported Request (UR) and discarded, as the poison rule mask bit (CX_FLT_MASK_UR_POIS) is not set in the Symbol Timer Register and Filter Mask Register 1. You can control the end result of a poisoned TLP by setting this mask bit, and having the poisoned TLP sent to your application.
- A locally terminated TLP with ECRC error detected is discarded in store-and-forward mode and an ECRC error reported only when the filter mask CX_FLT_MASK_ECRC_DISCARD bit is not set. For more information see ECRC Handling.
- Filter rules have no effect on received TLPs when 'DLLP ABORT' signal is asserted.

- If a completion of a non-posted request is not received within a completion timeout period, this request will be treated as a completion timeout, and a non-advisory error will be reported.
- For messages to be accepted and decoded, the incoming Message must be one of the valid Message types with the correct payload length based on *PCIe 3.0 Specification*. Valid Messages will be decoded and passed onto the SII interface as necessary.

See Error Handling for more details.

The filtering rules for an incoming Request TLP are affected by the configuration parameters (compile-time options), I/O signals (run-time options), and register values (run-time options).

NOTE

In many cases, the standard filtering rules may be 'masked' or ignored by setting the corresponding bit in the Symbol Timer Register and Filter Mask Register 1 and Filter Mask Register 2, which take their default values from the DEFAULT_FILTER_MASK_1 and DEFAULT_FILTER_MASK_2 configuration parameters.

48.3.4.2.2 Filtering Rules Based on TLP Type Defined in PCIe Specification

PCIe TLPs are categorized as Requests and Completions. The table found here describes the filtering rules for Request and Completion TLPs and the results of the core's filter.

If a received TLP passes all of the filter rules for Request and Completion TLPs, then it is considered to have no errors, and the TLP will be routed to the destination that is configured. Details on routing are provided in Receive Routing.

Notation of filter results:

1. UR = Unsupported Request

   CA = Completer Abort

   CRS = Configuration Request Retry Status

   SU = Successful

   UC = Unexpected Completion

   MLF = Malformed

   '-' = Filtering rule does not apply to TLP type

   MA = Master Abort

TA = Target Abort

48.3.4.2.2.1 EP MODE FILTERING RULES

Table 48-3. Result of Filtering Rules Applied to Request TLPs and Completion (CPL) TLPs: EP Mode

Filtering Rule	TLP Type					
	MRd IORd	MWr IOWr	CFG	MSG	CPL with UR/CA/CRS status	CPL with SU status
PowerState is not in D0.	UR	UR	SU	SU	UC	UC
Address is not within any configured Memory BAR or IO BAR if it is an IO request.	UR	UR	-	-	-	-
TLP header poison bit is set and the filter mask CX_FLT_MASK_UR_POIS bit is not set.	UR	UR	UR	UR	SU	SU
Address within a BAR that is configured to RTRGT0 and TLP DW length > 1.	CA	CA	-	-	-	-
MRd with lock and filter mask CX_FLT_MASK_LOCKED_RD_AS_UR bit is not set.	UR	-	-	-	-	-
The function number of a completer ID within a CFG request does not match an implemented function within the receiver device and the filter mask CX_FLT_MASK_UR_FUNC_MISMATCH bit is not set.	-	-	UR	-	-	-
Configuration type1 TLP request and the filter mask CX_FLT_MASK_CFG_TYPE1_REQ_AS_UR is not set.	-	-	UR	-	-	-
Application requests the core filter to return CRS by asserting signal app_req_retry_en.	-	-	CRS	-	-	-
Not Valid Message for EP device	-	-	-	UR/MLF	-	-
Illegal payload length of a message.	-	-	-	UR	-	-
Vendor MSG Type0 with filter mask CX_FLT_MASK_VENMSG0_DROP bit not set.	-	-	-	UR	-	-
Vendor MSG Type1 with r[2:0] to 3'b010 and {Bus#, Dev#, Func#} mis-match.	-	-	-	UR	-	-
TLP with ECRC error detected	CA	CA	CA	-	-	-
Requester ID mis-match	-	-	-	-	MA/TA	MLF
Requester TAG mis-match	-	-	-	-	MA/TA	MLF
TAG error (non-pad zero for reserved TAG bits	-	-	-	-	MA/TA	MLF
Byte Count Mismatch (PCIe Gen2)	-	-	-	-	MA/TA	UC/MLF

Table 48-4. Result of Filtering Rules Applied to Request TLPs and Completion (CPL) TLPs: EP Mode

Filtering Rule	TLP Type					
	MRd IORd	MWr IOWr	CFG	MSG	CPL with UR/CA/CRS status	CPL with SU status
Completion received with status of UR.	-	-	-	-	MA	-
Completion received with status of CA.	-	-	-	-	TA	-
Completion received with status of CRS	-	-	-	-	CRS	-
Completion received with CRS status and Completion is not a pending configuration request	-	-	-	-	MLF	-

A complete list of the filtering checks can be referenced at Symbol Timer Register and Filter Mask Register 1 (PCIE_PL_STRFM1) and Filter Mask Register 2 (PCIE_PL_STRFM2).

48.3.4.2.2.2 RC MODE FILTERING RULES

Table 48-5. Result of Filtering Rules Applied to Request TLPs and Completion (CPL) TLPs: RC Mode

Filtering Rule	TLP Type							
	MRd	MWr	CFG[1]	IO	MSG	CPL with UR/CA status	CPL with CRS status	CPL with SU status
Address does not satisfy any of the following conditions: 1. Within any configured Memory BAR. 2. Outside of the memory range AND prefetchable memory range as determined by the corresponding Base and Limit fields in the Type-1 header. 3. The filter mask CX_FLT_MASK_UR_OUTSIDE_BAR bit is set, which treats out-of-bar TLPs as Supported Requests and indicates a special application requirement	UR	UR	-	-	-	-	-	-
Native Core (no AHB/AXI bridge): FLT_Q_ADDR_WIDTH < 64, and any upper address bit (above bit position FLT_Q_ADDR_WIDTH-1) is set to '1'	UR	UR	-	UR	-	-	-	-
With AHB/AXI bridge: MASTER_BUS_ADDR_WIDTH = 32, FLT_Q_ADDR_WIDTH > 32,	UR	UR	-	UR	-	-	-	-

Table continues on the next page...

Table 48-5. Result of Filtering Rules Applied to Request TLPs and Completion (CPL) TLPs: RC Mode (continued)

and any upper address bit (above bit position MASTER_BUS_ADDR_WIDTH-1) is set to '1'								
TLP header poison bit is set and the filter mask CX_FLT_MASK_UR_POIS bit is not set.	UR	UR	UR	UR	UR	-	-	-
MRdLk request received and filter mask CX_FLT_MASK_LOCKED_RD_AS_UR bit is set, which indicates that customer prefer to filter out the MRdLk.	UR	-	-	-	-	-	-	-
CFG Request received and the filter mask CX_FLT_MASK_RC_CFG_DISCARD is not set	-	-	UR	-	-	-	-	-
IO Request received and the filter mask CX_FLT_MASK_RC_IO_DISCARD is not set	-	-	-	UR	-	-	-	-

Table 48-6. Result of Filtering Rules Applied to Request TLPs and Completion (CPL) TLPs: RC Mode

Filtering Rule	TLP Type							
	MRd	MWr	CFG[1]	IO	MSG	CPL with UR/CA status	CPL with CRS status	CPL with SU status
Vendor MSG Type0 with filter mask CX_FLT_MASK_VENMSG0_DROP bit not set.	-	-	-	-	UR	-	-	-
Not Valid Message for RC device	-	-	-	-	UR/MLF	-	-	-
TLP with ECRC error detected	CA	CA	CA	CA	-	-	-	-
Requester ID mis-match	-	-	-	-	-	MA/TA	-	MLF
Requester TAG mis-match	-	-	-	-	-	MA/TA	-	MLF
TAG error (non-pad zero for reserved TAG bits	-	-	-	-	-	MA/TA	-	MLF
Byte Count Mismatch	-	-	-	-	-	MA/TA	-	MLF
Completion received with status of UR.	-	-	-	-	-	MA	-	-
Completion received with status of CA.	-	-	-	-	-	TA	-	-
Completion received with CRS status and Completion is not a pending configuration request.	-	-	-	-	-	-	MLF	-

1. DM (in RC mode) should not expect to receive a CFG or IO request.

A complete list of the filtering checks see the Timer Register 1 and Filter Mask Register 2 in PCIe CTRL Memory Map/Register Definition .

48.3.4.2.3 Filtering Rules Not Defined in PCIe Specification

There are additional filtering rules that are designed to provide enhanced filter support for certain applications.

- Core to handle the received posted or non-posted requests with zero byte length. When a zero-byte request TLP is received, also called "flush" command, the core can drop[1] the zero- byte request. This is designed to support some applications that can not handle a zero-byte request. Applications can dynamically program a bit in the filter mask CX_FLT_MASK_HANDLE_FLUSH bit to turn on/off this rule. If the core is programmed to handle the flush, it will be the completer's task to return completion status.
- Core to detect oversize read request and return UR for the read request.Some applications may have a buffer limit and are not able to handle lengthy read requests. The core over-size read request detection rule can be turned on when an application can identify a maximum read request size that it can tolerate.

48.3.4.3 Receive Routing

48.3.4.3.1 EP Mode

The possible destinations of a posted or non-posted Request TLP are RTRGT1 interface, RTRGT0 interface and Core Discard (dropped[1] or terminated). By default:

- CFG requests are routed to RTRGT0 and then to CDM via LBC.
- BAR-matched MEM/IO requests are routed to RTRGT1.
- MSG requests are decoded internally, signalled on the SII interface and then terminated.

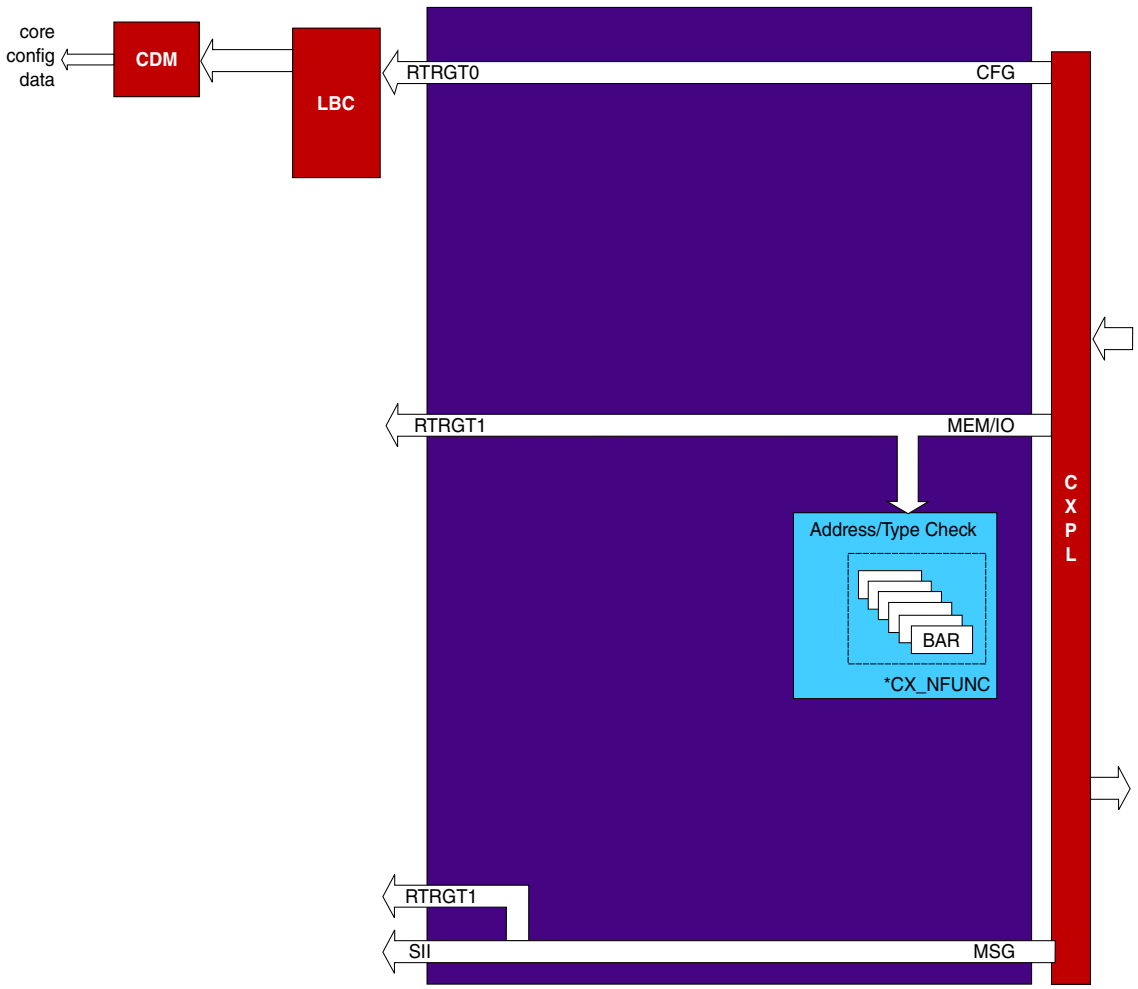


Figure 48-12. Default Request TLP Routing (assuming no TLPs with CA/CRS/UR completion status)

The possible destinations of a Completion TLP are RCPL interface, RBYP interface, RTRGT1 interface, and Core Discard.

In general, a TLP type that is configured as bypass will be sent to either the RBYP interface, or RCPL interface if it is a completion. A TLP type that is configured as a cut-through or store-forward will be sent to RTRGT1 interface.

Because the core supports three types of queue architecture (single queue, multiple queue, segmented buffer queue architecture) and three buffer modes (bypass, cut-through, store-forward mode), a configuration of the core receive queue structure will affect the destination of a received TLP.

NOTE

By default, all Configuration Requests that pass filtering go to RTRGT0 for configuration register access. However, the application can configure the core to direct certain

configuration TLPs to the RTRGT1 interface - see PCIe Wire Access (EP mode) for more details. The application is responsible for generating Completions for Configuration Requests that are routed to RTRGT1.

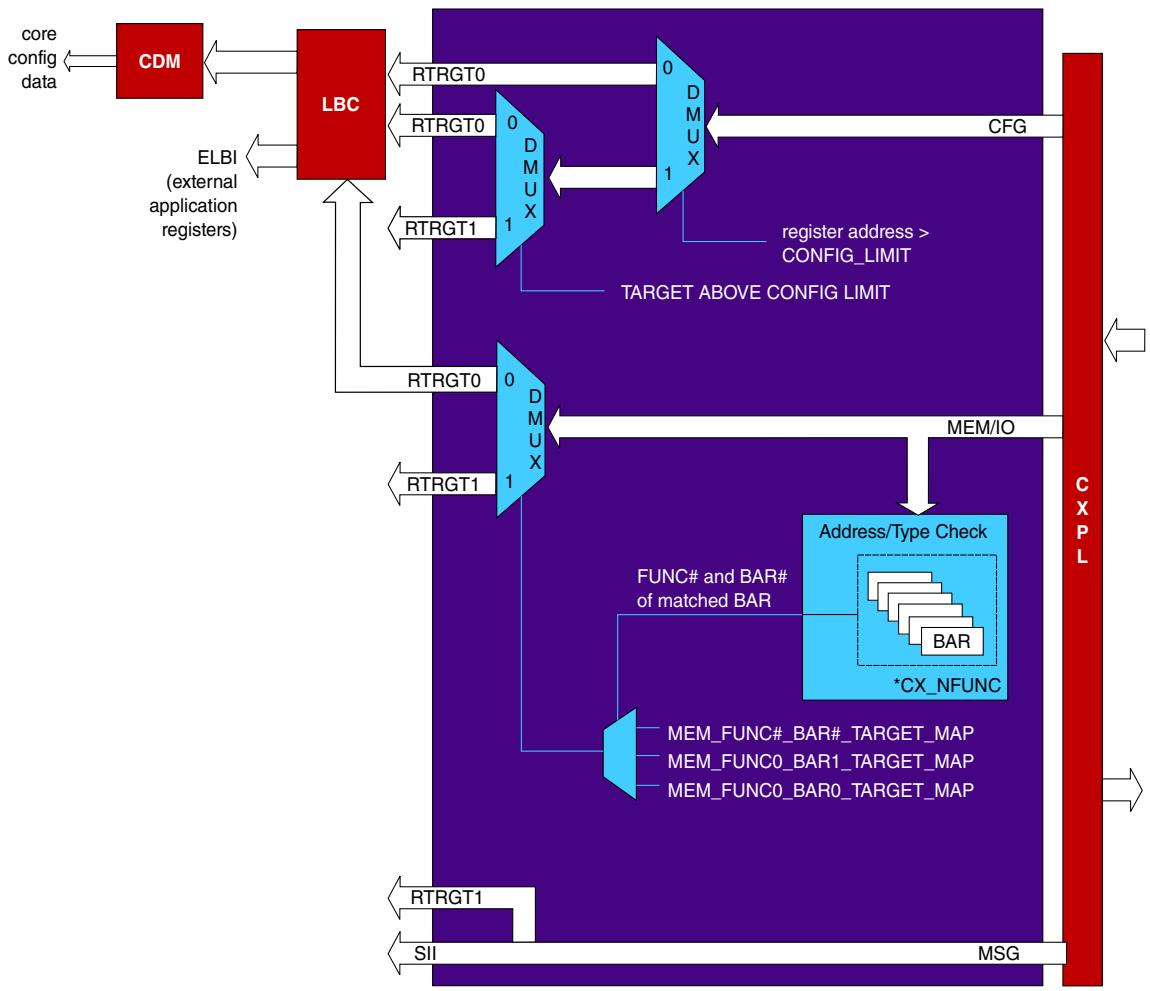


Figure 48-13. Configurable Request TLP Routing (assuming SC completion status).

48.3.4.3.2 RC mode

The possible destinations of a posted or non-posted Request TLP are RTRGT1 interface and Core Discard (dropped1 or terminated). By default:

- MEM requests outside of the memory range AND prefetchable memory range as determined by the corresponding Base and Limit fields in the Type-1 header, are routed to RTRGT1.
- MSG requests are decoded internally, signalled on the SII interface and then terminated.

- An RC does not expect to receive CFG or IO requests.
- BARs should be disabled and not used.

The possible destinations of a Completion TLP are RCPL interface, RBYP interface, RTRGT1 interface, and Core Discard.

In general, a TLP type that is configured as bypass will be sent to the RBYP interface. A TLP type that is configured as a cut-through or store-forward will be sent to RTRGT1 interface. Because the core supports three types of queue architecture (single queue, multiple queue, segmented buffer queue architecture) and three buffer modes (bypass, cut-through, store-forward mode), a configuration of the core receive queue structure will affect the destination of a received TLP.

48.3.4.3.3 ECRC Handling

The setting of the CX_FLT_MASK_ECRC_DISCARD bit (default value is 0) in the Symbol Timer Register and Filter Mask Register 1 can be used to prevent an ECRC contributing to a CA status and thereby preventing all associated downstream effects such as error handling.

By default, all incoming IO or MEM requests with UR/CA/CRS status will be dropped1/ terminated and an Advisory Non-Fatal Error is signalled. For Non Posted (NP) requests, a CPL with CA is generated. See Advisory Non-Fatal Error Messages.

If you set the DEFAULT_TARGET parameter to 'Forward' (default is 'Drop'), then all incoming IO or MEM requests with UR/CA/CRS status will not be dropped but will be forwarded to the application. Setting CX_MASK_UR_CA_4_TRGT1 (default is 0) at the same time, will suppress error reporting for TLPs (with UR/CA status) that are being routed to the application on AXI Bridge master.

A completion TLP with ECRC errors is only dropped by the RADM in the native core when CPL queue mode is store-forward and queue architecture is single or multiple queue. See Completion TLP Routing Rules for more details.

48.3.4.3.4 Request TLP Routing Rules

The next table shows the applicability of routing rules for Request TLPs, and indicates whether the destination is as stated by the rule when the conditions of the rule are met.

By default all error-free MSG requests are decoded internally, signalled on the SII interface and then terminated. To have the decoded message *also* sent to the application/ AMBA interface then see Routing of Received Messages to SII and optionally to

Application. When a MSG request is filtered with UR/CA/CRS status, the TLP is always terminated. Only MSG requests filtered with SC status, can potentially be forwarded to the application.

For a full analysis of what error conditions contribute towards an UR or CA status, see Receive Filtering. For example, an ECRC error contributes towards a CA status whereas an UR status can be generated by detecting the EP bit set in the TLP header or having an IO/MEM request not match against any of the BARs.

In many cases, the standard routing rules may be 'masked' or ignored by setting the corresponding bit in the Symbol Timer Register and Filter Mask Register 1 and îFilter Mask Register 2. For example, see Message Reception.

NOTE

RTRGT1 is the application interface and it is connected to the AXI bridge master interface.

Notation of routing results:

Yes = destination is as specified in rule when conditions of rule are met

no = destination is not as specified in rule even when conditions of rule are met

- = routing rule has no affect because it does not apply to TLP type

Table 48-7. Routing Rules for Request TLPs (EP Mode)

Routing Rule	MRd	MWr	CFG	IO	Vendor MSG Type0	Vendor MSG Type1	Other MSG
When a request is filtered with SU status, and is in BAR range, MEM_FUNC#_BAR#_TARGET_MAP parameter determines the destination.	Yes	Yes	no	Yes	-	-	-
When a request is filtered with UR/CA/CRS status, and the DEFAULT_TARGET parameter is 0, the TLP is dropped. For NP requests, a CPL is also generated.	Yes	Yes	Yes	Yes	Yes	Yes	Yes
When a request is filtered with UR/CA/CRS status, and the DEFAULT_TARGET parameter is 1, the TLP is dropped.	no	no	no	no	Yes	Yes	Yes
When a request is filtered with UR/CA/CRS status, and the DEFAULT_TARGET parameter is 1, the destination is RTRGT1 interface.	Yes	Yes	Yes	Yes	no	no	no
When a CFG request is filtered with SU status and the CFG register address is > CONFIG_LIMIT, TARGET_ABOVE_CONFIG determines the destination.	-	-	Yes	-	-	-	-

Table continues on the next page...

Table 48-7. Routing Rules for Request TLPs (EP Mode) (continued)

When a CFG request is filtered with SU status and the CFG register address is < CONFIG_LIMIT, RTRGT0 interface is the destination.	-	-	Yes	-	-	-	--
The TLP is dropped, when the filter mask CX_FLT_MASK_MSG_DROP bit is not set and the non-Vendor MSG is filtered with SU status.	-	-	-	-	no	no	Yes
The TLP is dropped, when the filter mask CX_FLT_MASK_VENMSG0_DROP bit is 0 and the VEN0 MSG is filtered with SC status.	-	-	-	-	Yes	no	no
The TLP is dropped, when the filter mask CX_FLT_MASK_VENMSG1_DROP bit is 0 and the VEN1 MSG is filtered with SC status. RTRGT1 interface is the destination when none of the previous rules are satisfied and the MSG is filtered with SU status.	- -	- -	- -	- -	no Yes	Yes Yes	no Yes

Table 48-8. Routing Rules for Request TLPs (RC mode)

Routing Rule	MRd	MWr	[1]CFG	[1]IO	Vendor MSG Type0	Vendor MSG Type1	Other MSG
When a request is filtered with SU status, and is not in BAR range, RTRGT1 is the destination.	Yes	Yes	no	Yes	-	-	-
[2]When a request is filtered with SU status, and is in BAR range, MEM_FUNC#_BAR#_TARGET_MAP parameter determines the destination.	Yes	Yes	no	no	-	-	-
When a request is filtered with UR/CA status, the TLP is dropped. For NP requests, a CPL is also generated.	Yes	Yes	Yes	Yes	Yes	Yes	Yes
The TLP is dropped, when the filter mask CX_FLT_MASK_MSG_DROP bit is 0 and the non-Vendor MSG is filtered with SU status.	-	-	-	-	no	no	Yes
The TLP is dropped, when the filter mask CX_FLT_MASK_VENMSG0_DROP bit is 0 and the VEN0 MSG is filtered with SU status.	-	-	-	-	Yes	no	no
The TLP is dropped, when the filter mask CX_FLT_MASK_VENMSG1_DROP bit is 0 and the VEN1 MSG is filtered with SU status. RTRGT1 interface is the destination when none of the previous rules are satisfied and the MSG is filtered with SU status.	- -	- -	- -	- -	no Yes	Yes Yes	no Yes

1. DM (in RC mode) should not expect to receive a CFG or IO request.

2. BARs are not normally used in RC application.

48.3.4.3.5 Completion TLP Routing Rules

The table found here shows the applicability of routing rules for Completion TLPs, and indicates whether the destination is as stated by the rule when the conditions of the rule are met.

In summary, under error conditions, CPLs are never dropped but always forwarded to the application except when:

1. 'suggested' completion or error status is any of the following:
   - UC
   - 'DLLP abort'
   - 'ECRC error'

   and

2. queue mode is store-forward and
3. queue architecture is single or multiple queue.

Table 48-9. Routing Rules for Completion TLPs.

Routing Rule	Filter Status of the Completion (CPL)	
	SC / UR / CA / CRS	UC / 'DLLP abort' / 'ECRC error'
Core Drop is the destination when queue mode is store-forward and queue architecture is single queue or multiple queue	no	yes
RCPL interface is the destination when queue mode is <any> and queue architecture is single queue or multiple queue	yes	yes
RBYP interface is the destination when queue mode is by-pass and queue architecture is segment buffer queue.	yes	no
RTRGT1 interface is the destination when queue mode is cut-through and queue architecture is segment buffer queue.	yes	no
RTRGT1 interface is the destination when queue mode is store-forward and queue architecture is segment buffer queue.	yes	no
Core Drop is the destination when queue mode is <any> and queue architecture is segment buffer queue.	no	yes

48.3.4.4 Receive Queuing

The core support three configurable queue architectures per VC: single queue, multiple queue, and segmented queue.

Each queue architecture supports three buffering modes: bypass mode, cut-through mode, and store-and-forward mode. The buffering mode is selectable for each TLP type: posted, non-posted and completion. The configurability is dependent on the queuing architecture.

The single queue architecture has one header buffer and one data buffer; and the header and data buffers are used as a single FIFO for buffering all posted, non-posted and completion TLP.

The multiple queue architecture has a single header and data buffer per posted, non-posted and completion TLP type.

The segmented queue architecture has one header buffer and one data buffer, but these two buffers are segmented by posted, non-posted and completion TLP type (versus single queue architecture).

For all queuing modes, RAM modules are either instantiated inside the top-level module of the core or connected externally, which is configurable.

48.3.4.4.1 Queuing Architecture

The queue architecture is specified by the user using the CX_RADMQ_MODE configuration parameter..

48.3.4.4.1.1 SEGMENTED-BUFFER RECEIVE QUEUE CONFIGURATION (CX_RADMQ_MODE=2)

The segmented-buffer queue architecture is designed for applications that want to enforce an ordering rule other than FIFO. This is the only queue architecture that can strictly adhere to the ordering rules of the *PCI Express Specification*. This queue architecture is relatively large in area. The segmented-buffer configuration uses a single memory module pair (header and data) for all TLP types and all VCs.

In the segmented-buffer configuration:

- The memory width is set automatically.
- The memory is divided into segments for Posted, Non-Posted, and Completion queues for each VC. The depth of each segment is set during hardware configuration,
- The operating mode is selected (bypass, cut-through, or store-and-forward) independently for each TLP type of each VC during hardware configuration. The operating mode (per TLP type and VC) can be controlled dynamically by writing to the Port Logic registers. If a TLP type is configured to be cut-through or store-and-forward, then it will be routed to the RTGT interfaces. If a TLP type is configured to be bypassed, then it will be routed to the radm_bypass interface. Once a Posted Request is configured in bypass mode, the application should not expect to send a Posted-write to the ELBI interface.

- The number of advertised credits is selected independently for each TLP type and each VC during hardware configuration. The number can be selected dynamically by writing to the Port Logic registers.
- The receive queue priority for VCs can be set to either strict priority (higher-numbered VCs have higher priority) or round robin during hardware configuration; and the priority at runtime can be set by writing to the Port Logic registers.
- The ordering rules for TLP types can be set during hardware configuration and at runtime by writing to the Port Logic registers. The choices are either strict priority (Posted first, Completion second, Non-Posted third) or priority determined according to ordering rules set forth in the PCI Express 3.0 Specification.

48.3.4.4.2 Queue Modes

The queue mode for each P/NP/CPL queue is specified using the RADM_P_QMODE_VC0 , RADM_NP_QMODE_VC0 and RADM_CPL_QMODE_VC0 configuration parameters.

It is possible to change the Queue Mode (during device setup by software) by writing to the appropriate queue control register.

BYPASS MODE (RADM_P/NP/CPL_QMODE_VC0=4)

Queues in bypass mode are completely bypassed.

In the case of aborted TLPs, the core asserts either radm_trgt1/cpl/bypass_dllp_abort or radm_trgt1/cpl/bypass_tlp_abort to the application. The application is responsible for rolling back any actions that were performed on behalf of the aborted TLP.

CUT-THROUGH MODE (RADM_P/NP/CPL_QMODE_VC0=2)

In cut-through mode, the queue presents data to the application as soon as the first data for a packet is placed into the queue.

In the case of aborted TLPs, the application is responsible for rolling back any actions that were performed on behalf of the aborted TLP. The radm_trgt1/cpl/bypass_dllp_abort and radm_trgt1/cpl/bypass_tlp_abort signals are presented to the application at the same time as the eot signal.

STORE-AND-FORWARD MODE (RADM_P/NP/CPL_QMODE_VC0=1)

In store-and-forward mode, only valid TLPs are forwarded to the application logic. Therefore, no rollback functionality is required by the application. All TLPs with radm_trgt1/cpl/bypass_dllp_abort or radm_trgt1/cpl/bypass_tlp_abort asserted will be dropped[1] by the receive queues of the core.

Flow Control credits are returned by the queue as packets are read out (even when the TLP was aborted and not presented to the application).

NOTE

Error handling by the application is dependent on the choice of queue mode. See Dependency upon Queue Architecture and Mode.

48.3.4.4.3 Order Enforcement

The ordering of TLPs within the same VC depends on the queuing architecture as follows:

- In the segmented-buffer configuration, either PCIe ordering rules or strict priority ordering are provided.
- In segmented buffer mode, you may select either strict VC priority (same as for single- and multi- buffer) or round robin.

For more information of packet ordering in relation to buffering/queue mode and architecture see:

- App Note: Order Enforcement Using the PCIe Core
- Inbound Order Enforcement for AXI Bridge

48.3.4.4.4 Queue to Port Mapping

Tables found here indicate which interfaces (ports) are used to deliver requests and completions depending on the queue architecture and buffer mode selected.

For Posted (P) and Non Posted (NP) TLP's, the destination (RTRGT1/RTRGT0) depends on the BAR setup, and if the TLP is of CFG type or not.

See Receive Routing for more details on routing for TLP's under error conditions.

QUEUE TO PORT MAPPING WHEN AHB/AXI BRIDGE PRESENT

RTRGT1 is connected to the AXI/AHB Bridge master interface (request) channel. RCPL and RBYP are connected to the AXI/AHB Bridge slave interface (response) channel. See Receive Routing for more details on routing for TLP's under error conditions.

Table 48-10. Segmented Queue Architecture (CX_RADMQ_MODE == 2) Queue to Port Mapping

Queue Mode	Posted	Non Posted	CPL

Table continues on the next page...

Table 48-10. Segmented Queue Architecture (CX_RADMQ_MODE == 2) Queue to Port Mapping (continued)

(RADM_P/NP/CPL_QMODE_VC0)	TLP Queue	TLP Queue	Queue
Store and Forward (1)	RTRGT1 / RTRGT0	RTRGT1 / RTRGT0	[1]RTRGT1
Cut Through (2)	n/a	n/a	n/a
Bypass (4)	n/a	n/a	RBYP

1. Use this feature if you want to maintain PCI Express ordering rules within the AHB/AXI Bridge module after packets have left the RADM receive queues in the native PCIe core. For more information see Ordering Enforcement Hardware Lock Feature.

48.3.5 Error Handling

An overview and additional information regarding error handling can be found here.

48.3.5.1 Error Handling Overview

Errors are classified into two levels:

- Correctable Error (CORR). This means that the PCIe core has a way of automatically handling the error. There is no loss of information. For example, Link CRC (LCRC) that is fixed by replaying the DLL.
- Uncorrectable Error (UNCORR). The PCIe core can not fix these and they are classified as:
- Fatal Error (FATAL). The link is not functioning correctly and may require a link reset.
- Non-Fatal Error (NONFATAL). The problem is not related to link operation.

The core implements the types of error handling found here.

- PCIe Baseline Capability. These reporting capabilities are a minimum set, and are required of all PCI Express devices. Error notification takes two forms:
- Messages sent to Root Complex (RC).
- Completion Status errors.

This also covers mapping of PCIe errors to legacy PCI generic error handling such as PERR# and SERR#. Many of the PCIe errors are mapped into the Status register in the PCI Compatible Configuration Space Header.

- PCIe Advanced Error Reporting (AER) Capability. Allows more sophisticated error reporting, control, masking and logging using the PCIe extended AER capability register structure.

The PCIe core supports Advisory reporting for both the Baseline and AER capabilities, which is the configurable with-holding of reporting for Non-Fatal errors (NONFATAL).

For an RC port, the reporting of most errors is internal to the root port. No external error notifications are generated. One exception to this (for example) is Unsupported Request (UR) completion status.

48.3.5.2 PCIe Baseline Capability

Reporting of errors is achieved by sending a notification to the RC (a CPL with UR/CA/CRS status for Non Posted requests, and optionally an error Msg).

The decision to send an error MSG is controlled by a complex set of associated control and status bits. The status is also logged in the Device Status Register for the following errors: Unsupported Request (UR), FATAL, NONFATAL and CORR.

MESSAGES SENT TO ROOT COMPLEX (RC).

Messages sent to the RC are of the ERR_CORR, ERR_NONFATAL, and ERR_FATAL types.

COMPLETION STATUS ERRORS.

Completion Status errors for Non Posted requests may be any of the following:

- Unsupported Request (UR)
- Configuration Request Retry Status (CRS)
- Completer Abort (CA)

REPORTING THROUGH THE DEVICE STATUS REGISTER

The PCI Express Capability Register Structure provides the following support for Baseline Error Reporting.

- Enable/disable error reporting (Device Control Register).
- Provide error status (Device Status Register) for:
- UR
- Correctable Error (CORR).
- Fatal Error (FATAL).
- Non-Fatal Error (NONFATAL).
- A method for software to force Link Retraining (Device Control Register).

48.3.5.3 Advanced Error Reporting (AER)

The decision to send an error MSG is controlled by a complex set of associated control and status bits. For more details, see the flow diagram in Error Detection.

AER allows more sophisticated error reporting, control, masking and logging using the optional extended AER capability register structure.

By default, AER is enabled (AER_ENABLE parameter), and may not be disabled unless you disable ECRC support in the core (CX_ECRC_ENABLE parameter).

AER REGISTERS

The AER registers are described in the Advanced Error Reporting Capability Registers section described in . All possible errors are enabled, masked and assigned a severity.

There are two sets of registers:

- Error Enable Register
- Error Severity Register
- Error Mask Register.

The Correctable set of registers handles (for example) errors arising from bad DLLPs or TLPs.

The Uncorrectable set of registers handles (for example) errors arising from UR, ECRC, Malformed TLPs, Buffer Overflow, UC, CA, Completion Timeout and Poisoned TLP.

SEVERITY PROGRAMMING

The Uncorrectable Error Severity register allows each uncorrectable error to be programmed to Fatal or Non-Fatal. The transmission of these error Messages by class (correctable, non-fatal, fatal) is enabled using the Reporting Enable fields of the Device Control register or the SERR# Enable bit in the PCI Command register. The Device Control Register and Command Register are described in PCIe CTRL Memory Map/ Register Definition.

The Uncorrectable Error Mask register and Correctable Error Mask register allows each error condition to be masked independently. If Messages for a particular class of error are not enabled by the combined settings in the Device Control register and the PCI Command register, then no Messages of that class will be sent regardless of the values for the corresponding mask register. If an individual error is masked when it is detected, its error status bit is still affected, but no error reporting Message is sent to the Root Complex, and the Header Log and First Error Pointer registers are unmodified.

48.3.5.3.1 Advisory Non-Fatal Error Messages

The PCIe core supports Advisory reporting which is the configurable with-holding of reporting for Non- Fatal errors.

- With Baseline Error Reporting, the core produces no error message.
- With AER, the core can instead, signal a non-fatal error with ERR_COR, which serves as an advisory notification to software.

It will always signal a fatal error with ERR_FATAL

UR/CA ADVISORY

The PCIe core generally sends a CPL with UR/CA status to signal a uncorrectable error for a Non-Posted Request. If the severity of the UR/CA error is non-fatal, the PCIe core will handle this case as an Advisory Non-Fatal Error.

By default, the PCIe core will signal the non-fatal error (if enabled) by sending an ERR_COR Message.

If AER is disabled (AER_ENABLE=0), the PCIe core sends no error Message for this case. Even though there was an uncorrectable error for this specific transaction, the PCIe core will handle this case as an Advisory Non-Fatal Error, since the Requester upon receiving the Completion with UR/CA Status is responsible for reporting the error (if necessary) using a Requester-specific mechanism.

UC ADVISORY

When the PCIe core receives an UC and the severity of the UC error is non-fatal, the PCIe core will handle this case as an Advisory Non-Fatal Error. By default, the PCIe core will signal the error (if enabled) by sending an ERR_COR Message.

If AER is disabled (AER_ENABLE=0), the PCIe core sends no error Message for this case.

48.3.5.4 Error Source Classification

The table found here indicates how some of the more common low level errors are classified.

Table 48-11. Possible Causes for Typical Errors

Error Type	Possible Cause
UR (Unsupported Request)	•Poisoned TLP (EP=1) •No BAR match •MRd length > max read request size

Table continues on the next page...

Table 48-11. Possible Causes for Typical Errors (continued)

UC (Unexpected Completion)	•TAG mismatch. •Requester ID (RID) mismatch.
CPL TimeOut	Remote device hung.
CA (Completion Abort)	ECRC
Malformed TLP (MLF)	Bad TLP header caused by bad link.
Buffer OverFlow	Credit miscalculation by some PCIe device.
Bad DLLP	LCRC

For a full analysis of what error conditions contribute towards an UR or CA status, see Receive Filtering.

NOTE

In many cases, the standard operation may be 'masked' or ignored by setting the corresponding bit in the Symbol Timer Register and Filter Mask Register 1, which take its default value from the DEFAULT_FILTER_MASK_1 configuration parameter.

NOTE

For example, A TLP with the poison bit set (EP=1) is considered an Unsupported Request (UR) only when the UR poison rule mask bit is not set.

48.3.5.5 Error Detection

Built into the core are all mandatory error detections, some optional error detections, and the error report mechanism based on the *PCI Express Specification*.

The core also has an option for the application to turn off the filter rules and perform its own error checking. For more details, see Receive Filtering.

The following general rules apply to all incoming TLPs:

- The core discards all incoming TLPs that have an invalid Type field. This TLP is treated as a 'TLP-ABORT'.
- A locally terminated TLP with ECRC error detected is discarded in store-and-forward mode and an ECRC error reported only when the filter mask CX_FLT_MASK_ECRC_DISCARD bit is not set.
- Filter rules have no affect on received TLP when 'DLLP-ABORT' signal is asserted.

- If a completion of a non-posted request is not received within a completion timeout period, this request will be treated as a completion timeout, and a non-advisory error will be reported. See Advanced Error Reporting (AER) for more details.
- 'DLLP-ABORT' is asserted as a result of one of two conditions:
- a. A data link layer error is detected (e.g. LCRC). A retry from a remote device will occur.
  b. UC or completion with ECRC error is detected. This condition is valid only when the application has configured the core with infinite credits. Because the completion buffer of the core or application has limited resources defined for expected completions, it is necessary to avoid overflowing the completion buffer by unexpected completions. Therefore 'DLLP-ABORT' is asserted to notify the core completion buffer (if completion is in store-forward mode) or application's completion buffer to rewind their buffer pointers when a completion with ECRC error or unexpected completion is detected.
- 'TLP-ABORT' is asserted as a result of one of three conditions:
- Malformed TLP
- UC
- ECRC

The figure below provides a flow chart of the error detections for a packet received from the PCIe wire.

Signals in this figure are internal to the core (except those prefixed with radm_*)

The signals radm_cpl_dllp_abort and radm_cpl_tlp_abort are intended for use with the Completion queue configured in either bypass or cut-through mode so that the core can notify the application of errors detected as the TLP is received.

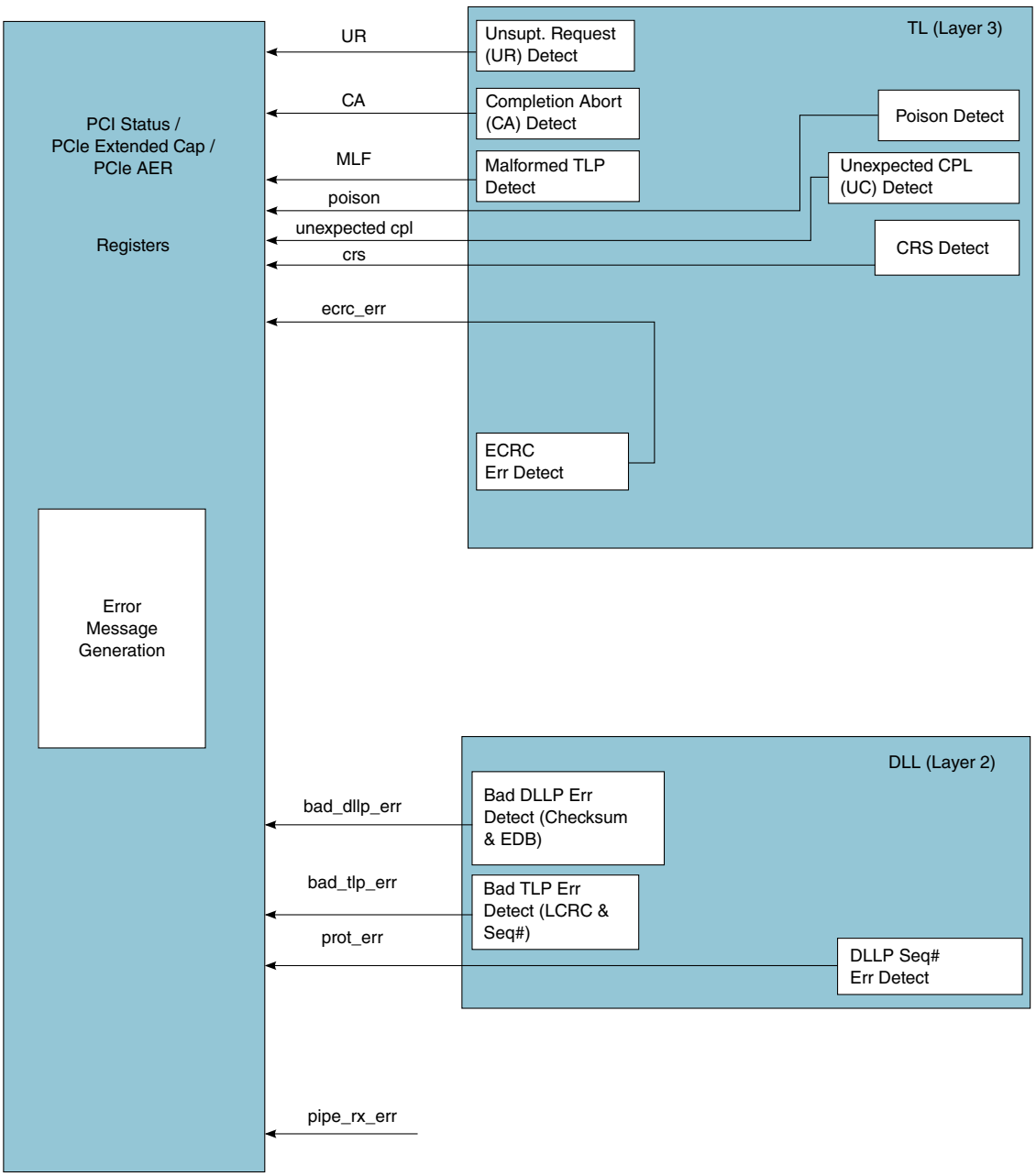


Figure 48-14. Flow Chart of Error Detections

48.3.5.6 Application Error Reporting Interface

The application may optionally generate an Error Message through either of the following methods:

- Directly at the application interface see Message Generation.

- It may instruct the PCI Express core to do so through the 'Application Error Reporting Interface'nsignals app_* .
- You must set the APP_RETURN_ERR_EN configuration parameter to enable the 'Application Error Reporting Interface'.

If you mask detection of Completion timeout errors, through setting the CPL_TIMEOUT_ERR_MASK configuration parameter, then the core will not automatically report Completion timeout errors. The application must check for Completion timeouts and report Completion timeout errors using the app_err_bus input signal.

NOTE

Your application may want to send an error message to the remote link partner if (for example) your application detected an error during the execution of an inbound/received Posted TLP, for example a MWr

48.3.5.7 Handling of General Errors with the AXI Bridge.

PCIe Advanced Error Reporting (AER) is not supported in the AXI bridge.

AER is only supported with respect to the native PCIe core. Errors detected by the AHB/AXI bridge are not reported as part of AER.

48.3.5.8 AXI

The slv_resp_err_map and mstr_resp_err_map bits are tied to 1. The corresponding CPL error will be SLVERR AXI Error Response. The corresponding CPL status error will be UR (Unsupported Request) PCIe CPL Status.

48.3.5.9 Handling of ECRC/LCRC Errors for IO/MEM with the AXI Bridge.

48.3.5.9.1 Link CRC (LCRC) - a correctable Error

When an LCRC error occurs on an inbound packet (request or completion), the core will automatically discard that packet and wait for the remote device to replay the packet.

The native core will signal (via radm_*_dllp_abort) to the AHB/AXI bridge that an LCRC error has occurred.

The AXI bridge will not signal an error to the application but will wait for the replayed packet to arrive.

48.3.5.9.2 End-to-end CRC (ECRC) - an Uncorrectable Non-Fatal Error

REQUESTS

By default, a request TLP with ECRC errors is dropped by the RADM filter in the native core and an Advisory Non-Fatal Error is signalled. For Non Posted (NP) requests, a CPL with CA is generated. See Advisory Non-Fatal Error Messages.

If you set the DEFAULT_TARGET and CX_MASK_UR_CA_4_TRGT1 parameters to 1 (default values are 0), then, a request TLP with ECRC errors will not be dropped but will be forwarded (without any error reporting) to the application on AXI bridge master.

See ECRC Handling for more details.

COMPLETIONS

A completion TLP with ECRC errors is only dropped by the RADM in the native core when CPL queue mode is store-forward and queue architecture is single or multiple queue. However, for the AHB/AXI bridge, when the queue architecture is single or multiple queue, the CPL queue mode is restricted to bypass. Therefore a completion TLP with ECRC errors is always forwarded to the AHB/AXI bridge.

When the AXI/AHB bridge receives a CPL with ECRC errors, it will not transmit it to the application through the AHB/AXI master. Instead, the bridge will wait for the subsequent 'completion timeout' (see Received Completion TLP Processing) generated by the core. It is this timeout that will cause the bridge to issue an ERROR response to the application on each beat of the burst corresponding to the original AHB/AXI request.

48.3.6 Messages

Information found here describes the processing of messages through the core.

NOTE

For a proper understanding of Messages you should be familiar with Message Request Rules of the PCI Express Base Specification.

Similar to MWr, messages (Msg/MsgD) are Posted transactions. The 8-bit 'Message Code' field defines what class of message the TLP is. Some examples of typical message classes are given in the following table.

Table 48-12. Some Message classes based on the Message code

Message Code [7:0]	Message Class	TLP Type	Note
0001_xxxx	Power Management	Msg	
0010_0xxx	Legacy PCI Interrupt	Msg	Assert/deassert for each of INT A/B/C/D.
0011_00xx	Error Signalling	Msg	ERR_CORR, ERR_NONFATAL, ERR_FATAL are encoded using 30h, 31h, 33h.
0111_11xx	Vendor Defined	Msg / MsgD	
Other classes (used by PCIe core) include Locked Transaction, Slot Power Limit.			

NOTE

Message Signalled Interrupts (MSI/MSI-X) are not messages (Msg/MsgD) but MWr TLPs. See Interrupts for more details.

48.3.6.1 Message Generation

Messages that are transmitted by the PCI Express core can potentially be derived from the following eight sources. Referring to the circled numbers in the following diagrams, outbound messages can be created either by:

the core automatically as follows:

- 'Power Management' messages.
- 'Error Signalling' messages.

or

the customer application as follows:

- Direct supply of message TLPs at AXI bridge master. An internal or external internal address translation unit (ATU) can convert IO/MEM TLPs to MSG TLPs.
- 'Locked Transaction' messages through the 'SII Message' interface [RC mode]. 'Legacy PCI Interrupt' messages through the 'SII Interrupt' interface.
- 'Error Signalling' messages through the 'SII Transmit Control' interface (app_err* I/O).

or

the host/client software as follows [RC mode]:

- Triggering the sending of 'Slot Power Limit' messages by writing to the Slot Capabilities Register.

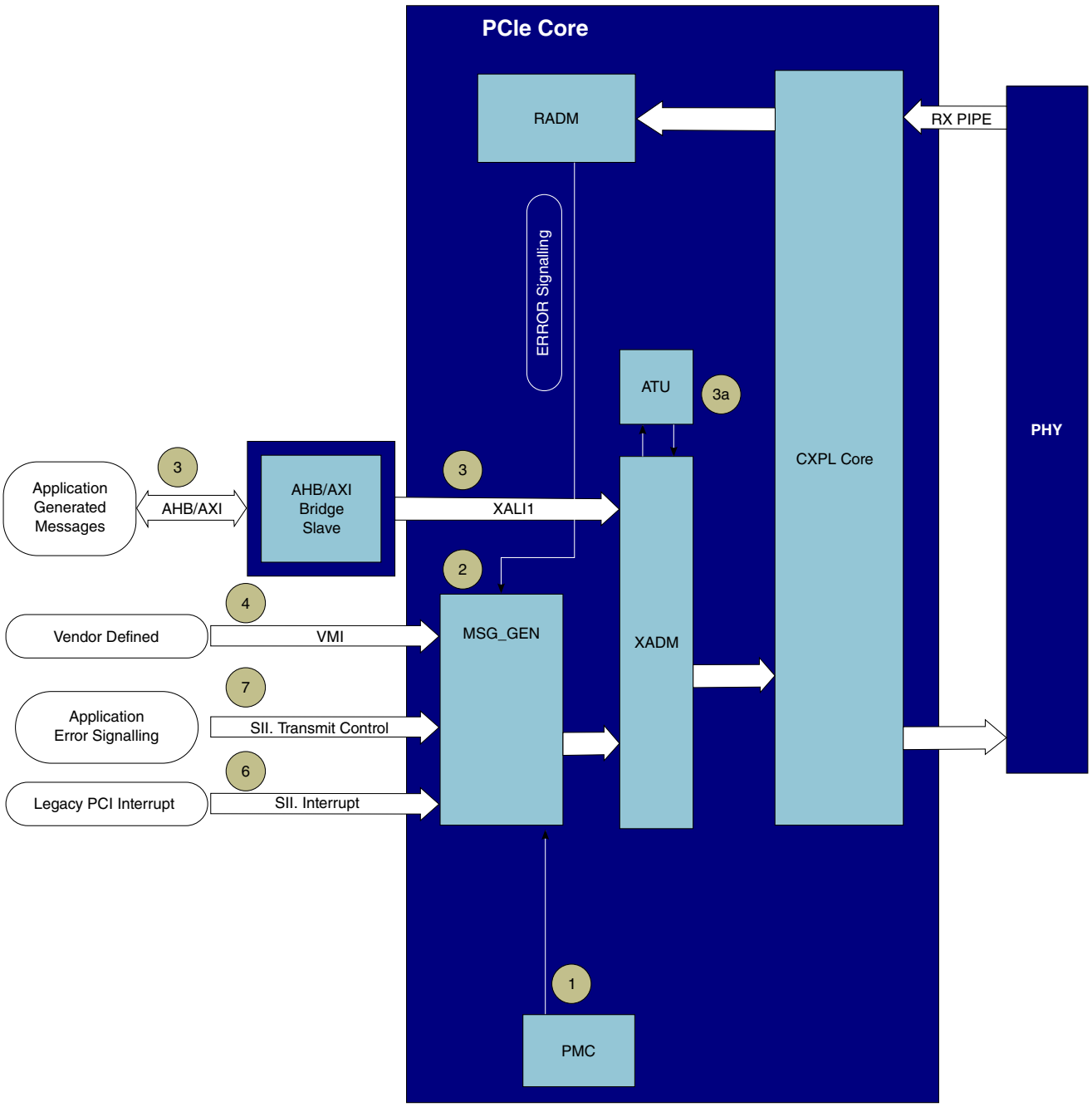


Figure 48-15. Message Transmission: EP mode

See Terms and Abbreviations for definitions of acronyms used for block and interface names.

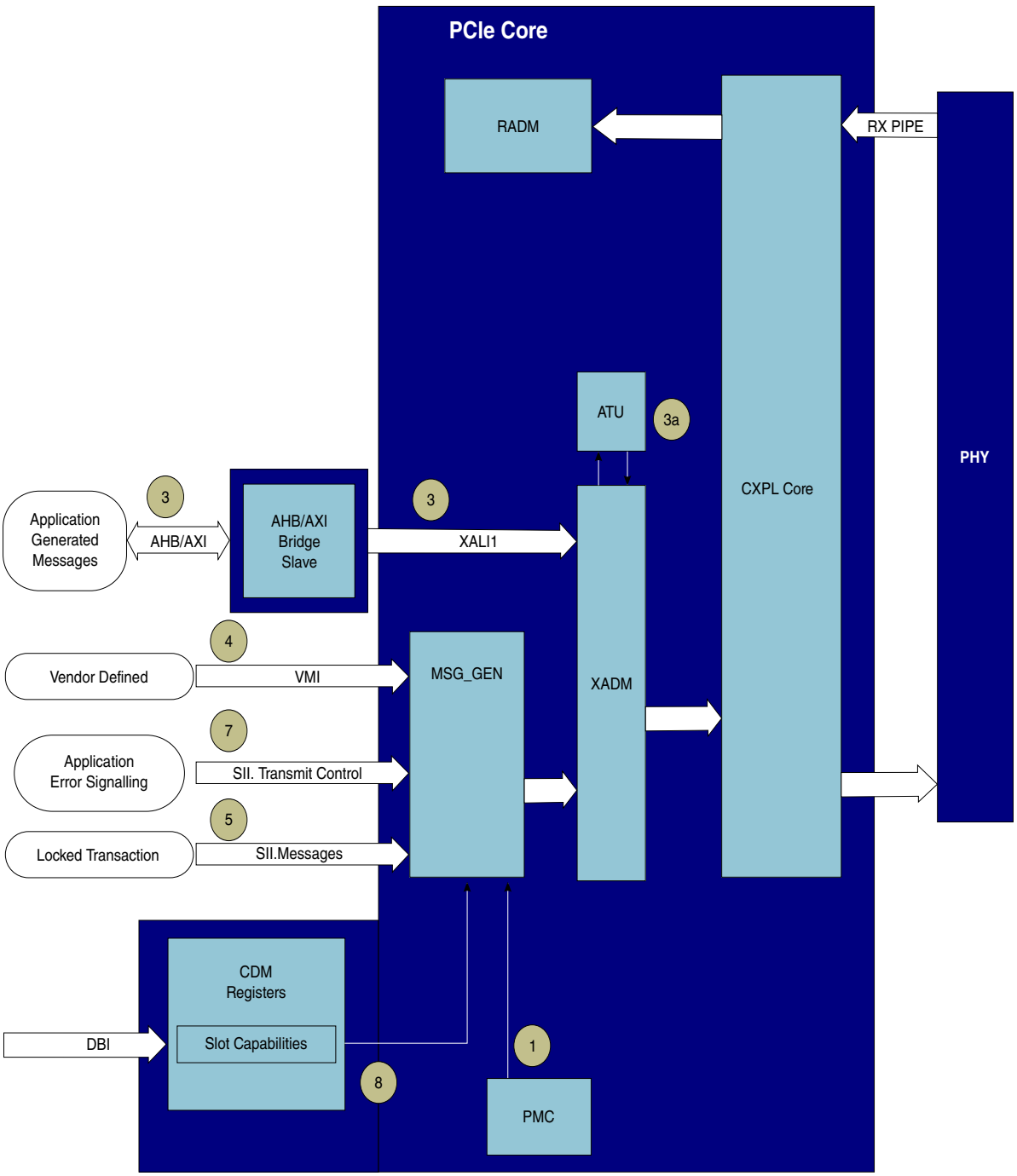


Figure 48-16. Message Transmission: RC mode

Table 48-13. Message Transmission. The 'Index' refers to the numbers in the previous diagrams

Index	Message Source (Type)	EP Mode	RC Mode

Table continues on the next page...

Table 48-13. Message Transmission. The 'Index' refers to the numbers in the previous diagrams (continued)

1	Power Management controller in the core (Msg).	PM_PME[1]	PME_Turn_Off[2] See Power Management.
2	Error Signalling inside the core (Msg).	COR_ERR / ERR_NONFATAL / ERR_FATAL. See Error Handling for more details.	n/a
3	Direct Supply of any class of Message (Msg/MsgD).	XALI0/1/2 or AHB/AXI See Application Msg/MsgD Programming Examples for details on how to generate a message at the XALI0/1/2 (or AHB/AXI) interfaces.	
3a	Indirect Supply of any class of Message (Msg/ MsgD).	See Outbound iATU Operation for more details on generating Msg/MsgD from MWr/IOWr using an internal or external address translation unit (ATU).	
5	Locked Transaction (Msg).	n/a	Unlock Message, triggered by Root Complex application logic via the app_unlock_msg pin.
6	Legacy PCI Interrupt (Msg).	'SII Interrupt' pins sys_int and dp_intx (see PCI Legacy Interrupt).	n/a
7	Error Signalling from the application (Msg).	The core generates Error Signalling Messages in response to application requests on the SII app_err* I/O . It is also possible to generate Error Messages via the client interfaces. See items 3 and 3a in this table.	
8	Slot Power Limit (Msg).	n/a	Set_Slot_Power_Limit Support Message, triggered by writing to the Slot Capabilities Register via the DBI.

1. Triggered by your EP application through the outband_pwrup_cmd or apps_pm_xmt_pme pin.

2. Triggered by your RC application through the apps_pm_xmt_turnoff pin.

3. MsgD not possible on VMI. See Vendor Defined Message (VDM) Generation.

48.3.6.1.1 Vendor Defined Message (VDM) Generation

VDMs can be generated by your application using any of the following methods (numbered 4, 3 and 3a in Table 48-13).

- Direct supply of IO/MEM TLPs AXI bridge master to be converted to VDM by either of the following.
- The internal Address Translation Unit (iATU) can convert IO/MEM TLPs to VDM TLPs.

48.3.6.1.1.1 Application Msg/MsgD Programming Examples

The tables found here enumerate the different ways your application can generate Msg and MsgD TLPs.

NOTE

xATU = "External Address Translation Unit" and iATU = "Internal Address Translation Unit (iATU)".

Table 48-14. Msg (message without payload) Generation Methods

Application Interface	Description	Application I/O Signals
AXI	Direct Supply using a Msg transaction.	slv_awmisc_info[4:0]='MSG" slv_wstrb[3:0]='0000'
	Indirect Supply (iATU) using a MWr[1] transaction. The iATU needs to be configured to translate MWr to Msg TLPs.	slv_awmisc_info[4:0]='MEM' slv_wstrb[3:0]='0000'

1. Or IOWr

NOTE

xATU = "External Address Translation" and iATU = "Internal Address Translation (iATU)".

Table 48-15. MsgD (Vendor Specific Message with payload) Generation Methods

Application Interface	Description	Application I/O Signals
AXI	Direct supply using an MsgD transaction.	slv_amisc_info[4:0]='MSG'
	Indirect Supply (xATU) using a MWr transaction.	slv_amisc_info[4:0]='MEM' xtranslated_enable =1 xtranslated_addr_in_d[8]=1 xtranslated_addr_in_d[7:0]=Message Code xtranslated_type_in_d = 10xxx
	Indirect Supply (iATU) using a MWr[1] transaction. The iATU needs to be configured to translate MWr to MsgD TLPs.	slv_amisc_info[4:0]='MEM'

1. Or IOWr

48.3.6.1.2 AHB/AXI Message Address and Size Limitations

Limitations existing in the current implementation of the AHB/AXI bridge module can be found here.

NOTE

- Vendor Messages must not be decomposed in the inbound or outbound direction.
- You must ensure that Vendor Defined Messages generated by your application or remote link partner, do not trigger decomposition as described in AXI Decomposition Rules.

Table 48-16. Processing of Inbound Messages when AHB/AXI and PCIe Core Address Widths are different

AHB/AXI Address Bus Width MASTER_BUS_ADDR_WIDTH	PCIe Core Address Bus Width FLT_Q_ADDR_WIDTH	Notes
32	64	The upper 32 bits of the data field (bytes[1] 8-11) are not forwarded by the bridge master.
64	32	The upper 32 bits of data field (bytes 8-11) will be forced to '0'.

48.3.6.2 Message Reception

The PCI Express core can receive the following types of messages. The index in the first column refers to the circled numbers in the following diagrams.

.

Table 48-17. Message Reception. The 'Index' refers to the numbers in the following diagrams

Index	Message Source (Type)	EP Mode	RC Mode
1	Power Management (Msg).	PME_Turn_Off See Power Management.	PM_PME PME_TO_Ack
1a	Slot Power Limit (Msg).	Set_Slot_Power_Limit Support Message.	n/a
2	Error Signalling from downstream component (Msg).	n/a	COR_ERR / ERR_NONFATAL / ERR_FATAL.
3	Vendor Defined (Msg/MsgD).		
4	Locked Transaction (Msg).	Unlock Message.	n/a
5	Legacy PCI Interrupts from downstream devices (Msg).	n/a	See PCI Legacy Interrupt.

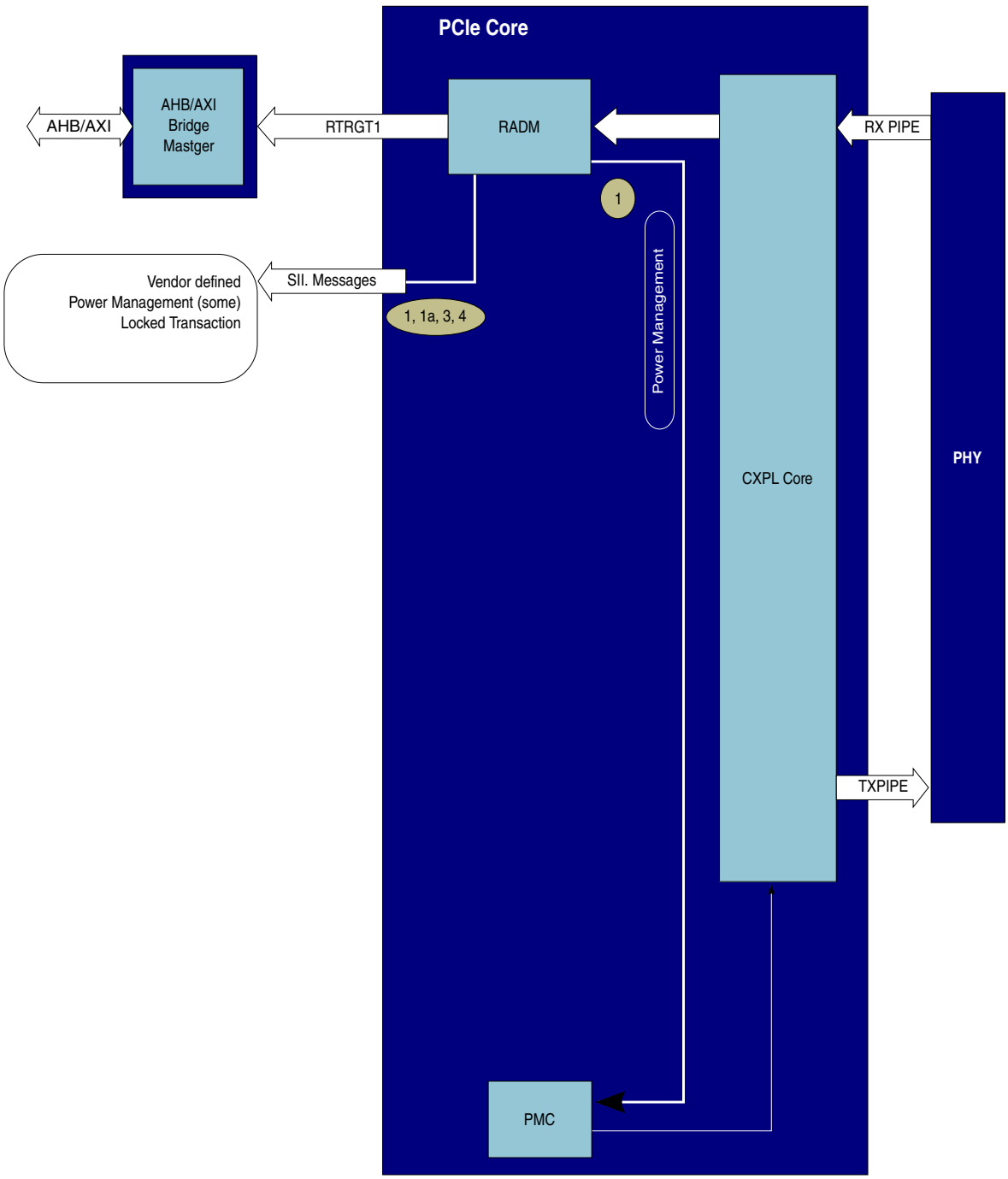


Figure 48-17. Message Reception: EP mode

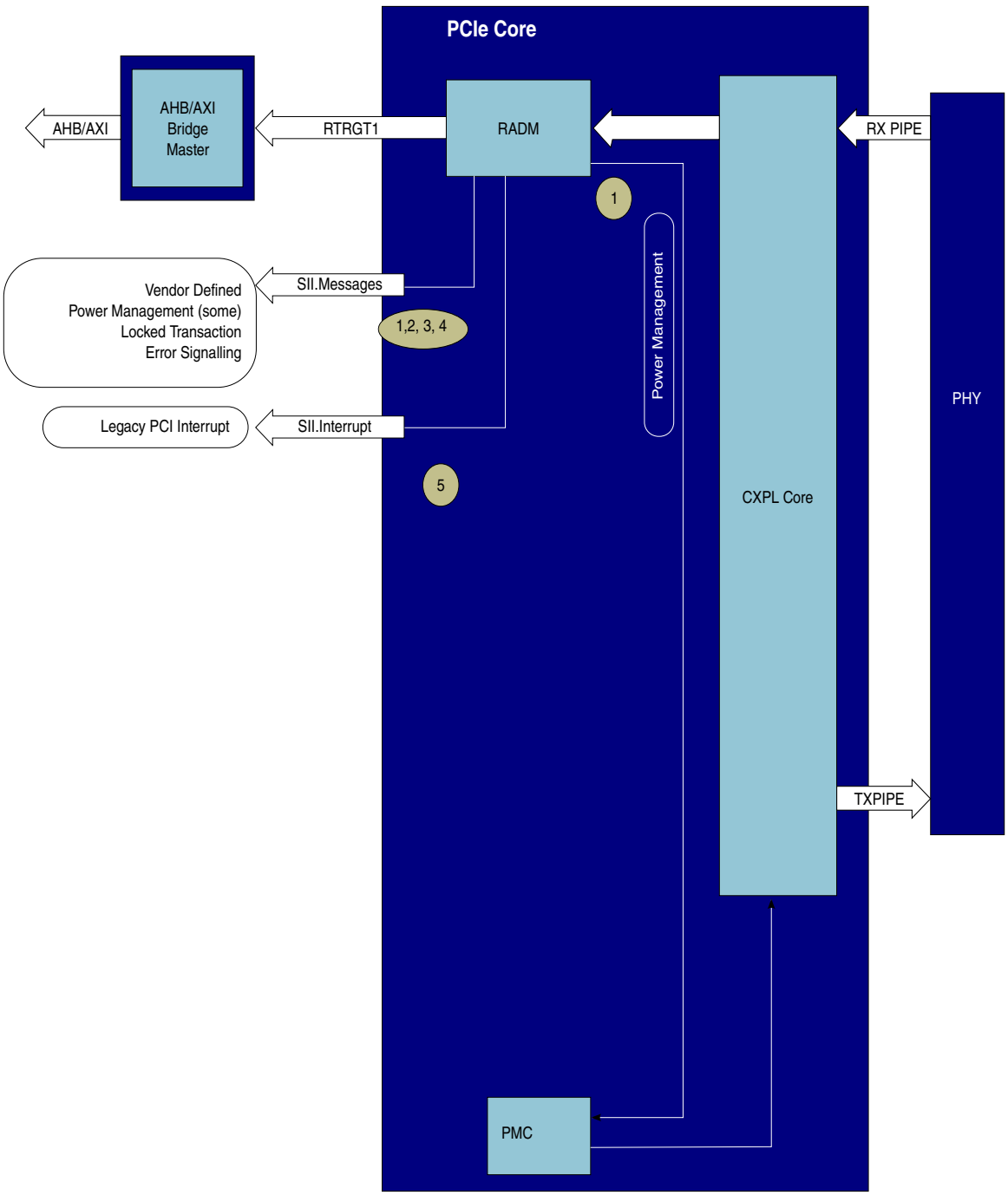


Figure 48-18. Message Reception: RC mode

See Terms and Abbreviations for definitions of acronyms used for block and interface names.

48.3.6.2.1 Message Reception IO Interfaces

The RADM filter provides a Message interface - that is grouped as part of the System Information Interface (SII) - to handle the Message TLPs received from the upstream component. The RADM filter processes the Message and decodes the header before sending it to the application logic on the SII.

48.3.6.2.2 Routing of Received Messages to SII and optionally to Application

The RADM filter processes *every* received message and decodes the header before sending it to the application logic on the System Information Interface (SII).

In addition, power management messages are processed by the PCIe core Power Management Controller (PMC).

By default, all received messages are dropped[1] (serviced internally) and *not* passed to the application on AXI bridge master.

To have all decoded messages *also* sent to the application interface (AXI bridge master), the register fields must be set to '1'. These registers allow you to override any decisions (regarding MSG routing) made at configuration time by the FLT_DROP_MSG, DEFAULT_FILTER_MASK_1 and DEFAULT_FILTER_MASK_2 configuration parameters .

Table 48-18. Controlling the Routing of Received Messages

Register	Bit	Function	Default Value
Filter Mask Register 1	29	Mask the dropping of Non-Vendor Messages 0: Drop 1: Do not drop	DEFAULT_FILTER_MASK_1[13] = ! FLT_DROP_MSG = 0
Filter Mask Register 2	0	Mask the dropping of Vendor Type 0 Messages 0: Drop[1] 1: Do not drop	DEFAULT_FILTER_MASK_2[0] = 0
Filter Mask Register 2	1	Mask the dropping of Vendor Type 1Messages 0: Drop 1: Do not drop	DEFAULT_FILTER_MASK_2[1] = 0

1. Vendor TYPE0 Messages are dropped with UR error reporting.

For the masking (of the dropping) of Vendor messages, it is not possible to differentiate between 'Vendor Message without Payload (Msg)' and 'Vendor Message with Payload (MsgD)'.

Full details of the Filter Mask Registers are at Symbol Timer Register and Filter Mask Register 1and Filter Mask Register 2.

When a MSG request is filtered with UR/CA/CRS status, the TLP is always dropped. Only MSG requests filtered with SC status, can potentially be forwarded to the application on AXI bridge master.

48.3.6.2.3 Accessing Header and Payload Fields of Received Messages

Format of Received Messages on the Application RTRGT1 Interface

In addition to normal TLP header information, the last two DWORDS of the message header (bytes[1] 8-15) are presented on radm_trgt1_addr[FLT_Q_ADDR_WIDTH-1:0] when radm_trgt1_hv is asserted.

Format of Received Messages on the Application AHB/AXI Master Interface

In addition to normal TLP header information, the last two DWORDS of the message header (bytes[1] 8-15) are presented on the master address bus as follows:

- AHB/AXI address[31:0] = 4th DWORD = bytes 12-15
- AHB/AXI address[63:32] = 3rd DWORD = bytes 8-11

The TYPE field of mstr_awmisc_info/mstr_req_misc_info indicates the type of message TLP received. The remote link partner should not use the last DWORD of a Msg/MsgD header (bytes[1] 12-15). If the last DWORD must be used, then the two lower bits (byte15) must be always set to 00b. If these two bits are not zero, then the AHB/AXI master interface (when it forwards the message packet to the AHB/AXI fabric) will initiate a burst of 8-bit or 16-bit transactions.

Messages Without Payload Restriction on AHB

Messages without Payload (Msg) cannot be forwarded to the AHB bridge module master interface, as there is no concept of 'write strobes' in AHB to support null/empty write packets. Therefore it is only possible to access (parts of) the received Msg (message without payload) over the SII.

Format of Received Messages on the SII

Not all of the message TLP fields are sent to the SII. For some messages, only an indication that it has been received is signalled. For Vendor Defined messages (Msg or MsgD), the 3rd DWORD (bytes[1] 8-11) is sent to radm_msg_payload[31:0] but the fourth DWORD (bytes 12-15) is not presented at all. Furthermore, for Vendor Defined messages with payload (MsgD), the payload is not presented on SII.

Therefore, to access this missing information (if required), it is necessary to unmask the dropping[2] of Vendor Messages and have them sent to the application interface (AXI bridge master) in order to access the payload. This can be problematic for AHB as it is not possible to differentiate between 'Vendor Message without Payload (Msg)' and 'Vendor Message with Payload (MsgD)' for the masking (of the dropping) of Vendor messages. To avoid, this problem, program the remote link partner to not send Vendor Defined messages without payload (Msg), to an AHB connected end point.

48.3.7 Interrupts

Information found here describes the processing of interrupts through the core.

48.3.7.1 Interrupts Overview

The application logic in a PCI Express Endpoint may use one of three methods to signal an interrupt:

PCI legacy interrupt

PCI includes up to four virtual interrupt wires, referred to as INTA, INTB, INTC, and INTD. These wires are shared by all the PCI devices in the system. PCI Express emulates this capability by providing Assert_INTx and Deassert_INTx Message packets sent through the PCI Express serial Link.

MSI

A PCI Express Endpoint may signal an MSI by sending a standard PCI Express Posted Write packet towards the Root Port. The packet must contain a specific address and one of up to 32 data values. The varying data values, and the address value provide more detailed identification of interrupt events than legacy interrupts.

The PCI Express Cores support optional MSI per-vector masking (PVM).

MSI-X

An MSI-X interrupt is identical to an MSI, except that an Endpoint may use one of up to 2048 address and data pairs in the MSI-X Posted Write packet. Endpoints with MSI-X capability also include application logic to mask and hold pending interrupts, as well as a memory table for the address and data pairs. The large number of address values available to each Endpoint allows MSI-X Messages to be routed to different interrupt consumers in a system, as compared to the single address available to MSI packets.

Root Ports cannot send MSI-X packets. In complex systems, MSI-X packets could be routed to devices other than the RC, including other Endpoints, based on the multiple address/data pairs available.

Support of legacy interrupts and/or MSI/MSI-X may be required for backward compatibility.You may configure your core and application logic to support all three types of interrupts. However, you may only use one of these capabilities at a time. When host software clears the MSI Enable bit, you may only use legacy interrupts. When host software sets the MSI Enable bit, you may only use MSI. If if host software enables MSI or MSI-X, legacy interrupts are automatically disabled. Functionality is undefined if both MSI and MSI-X are enabled.

48.3.7.1.1 PCI Legacy Interrupt

NOTE

Legacy Interrupt delivery are signalled using special Msg TLPs. For more details see Messages.

The PCI Express Endpoint Core provides an input pin (sys_int[NF-1:0]) per function, so that application logic can assert or deassert a legacy interrupt.

The Core automatically maps assertion/de- assertion edges on the input pin to PCI Express Assert or Deassert messages. Root Ports decode received Assert/Deassert messages into pulses. Refer to System Information Interface (SII) for additional information.

Multifunction Support

A single-function Endpoint always uses INTA. However in a multi-function Endpoint, you may choose which virtual interrupt is used for each function by setting the initial value of the Interrupt Pin configuration register. In a multi-function Endpoint, each function has its own interrupt input pin (sys_int[NF-1:0]). Each function's Interrupt Pin register determines which legacy interrupt Message the function uses (INTA, INTB, INTC, or INTD).

Ordering Considerations

You may wish to guarantee that a legacy interrupt Message is sent after a data packet. In that case, do not assert the interrupt input pin until after the data packet's header is accepted by the core's transmit client interface.

In a complex PCI Express system, which include Switches and/or multiple Virtual Channels, you cannot guarantee that the interrupt Message will arrive after a data packet, unless the data packet uses the same Traffic Class as the legacy interrupt packet (Traffic Class 0).

Deassertion of Interrupts

The application needs to deassert the virtual interrupt inputs if system software has disabled interrupts or if the PCI Express link has been placed in a low power state.

The Core does automatically send a Deassert Interrupt Message when software disables interrupts. However, the application must eventually deassert the virtual interrupt before it can send a new interrupt because the Core requires a rising edge on the virtual interrupt signal to generate a new Assert Interrupt Message.

The Core does not automatically send a Deassert Interrupt Message when the power state changes.

48.3.7.1.2 MSI

Message Signalled Interrupts (MSI) are not messages (Msg/MsgD) but Memory Write (MWr) TLPs. They are indistinguishable from normal MWr's apart from the target address used in conjunction with the MSI Capability Structure. PVM (Per Vector Masking) is supported with MSI.

The PCI Express Core automatically builds an MSI packet for your application (if MSI is enabled) whenever requested by your application logic.

A simple handshake is required. The MSI interface is used only in the DM core in EP mode.

The Core informs the application logic whether MSI interrupts are enabled, and how many MSI data vectors have been allocated by system software.

Before performing the handshake, application logic must assert Traffic Class, and MSI vector information on the Core input pins. The Core inserts the Traffic Class into the MSI packet. It also merges the MSI vector number into the MSI packet data field.

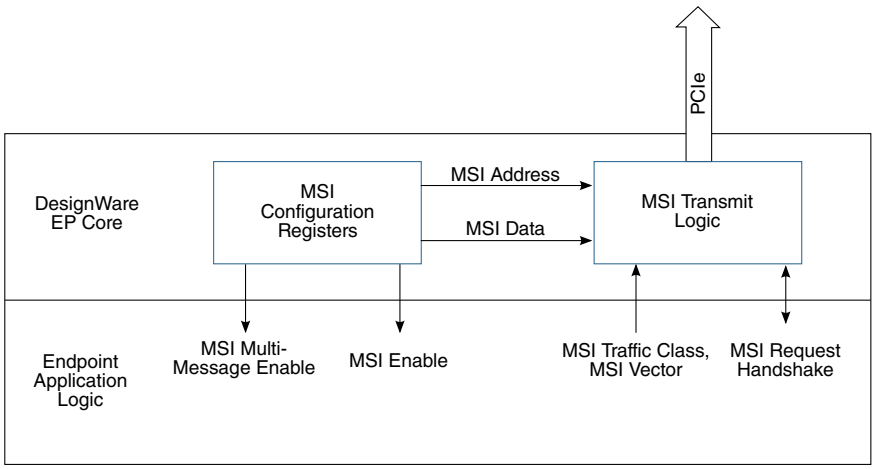


Figure 48-19. MSI Message Passing

When an Endpoint sends an MSI Message, the Message packet is routed by address in the same way as any other PCIe Posted Write packet. In most systems, these packets will be routed to the Root Port and sent to Root Port application logic. The Root Port application logic recognizes the Posted Write packet by its MSI address and handles the interrupt in hardware or software. The data field is also available in the packet to further define the interrupt.

Multifunction Support

Each function in a multi-function device has its own configuration space, and therefore, its own MSI output controls. A multi-function core adds a function number to the MSI inputs shown in the following figure.

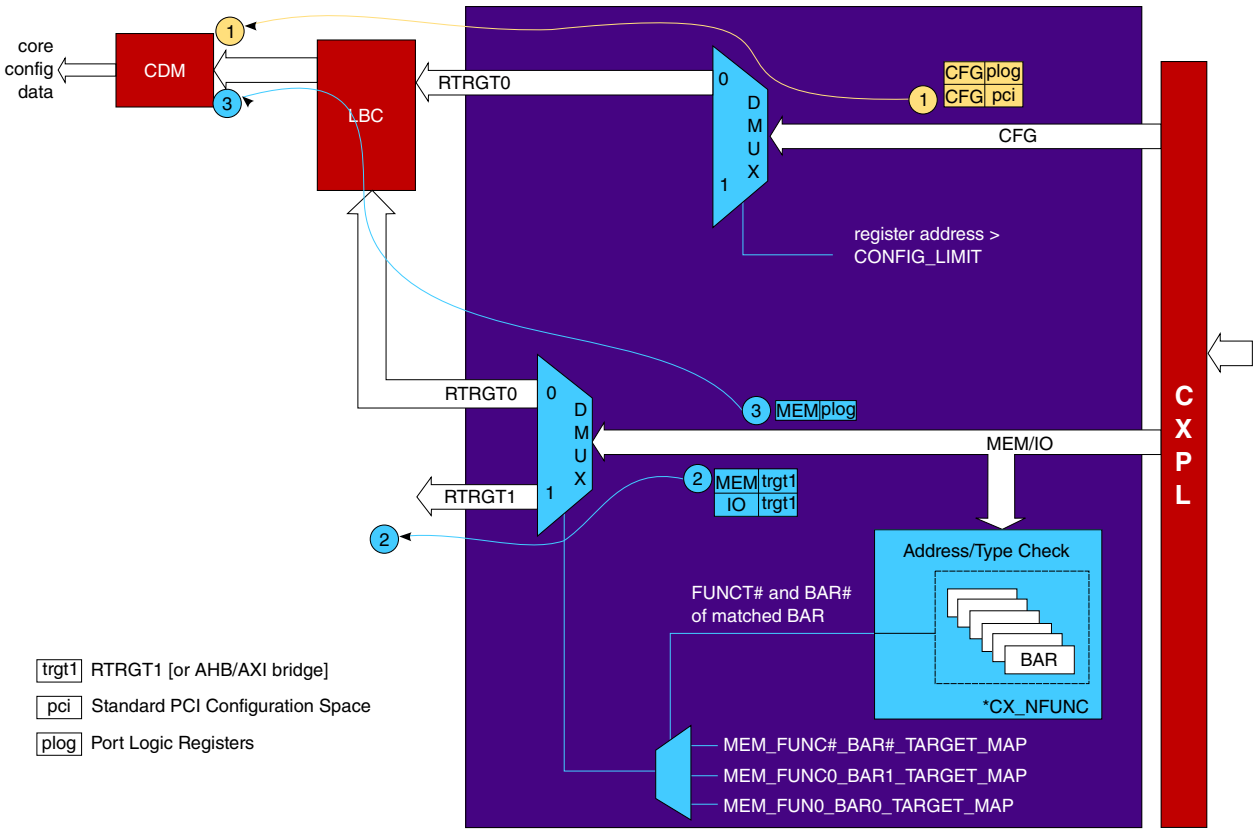


Figure 48-20. Typical use model

Ordering Considerations

You may wish to guarantee that an MSI is sent after a data packet. By setting the MSI packet's Traffic Class to the same value as an earlier data packet, you can guarantee that the MSI packet will always arrive at its destination after the data packet.

NOTE

The PCIe core sets the first byte enable (FBE) to 4'b1111 when it generates an MSI request, even though only the first two bytes of the payload are strictly valid/needed.

48.3.7.1.3 MSI-X

Message Signalled Interrupts (MSI-X) are not messages (Msg/MsgD) but Memory Write (MWr) TLPs. They are indistinguishable from normal MWr's apart from the target address used in conjunction with the MSI-X Capability Structure.

The PCI Express Core automatically builds an MSI packet for your application (if MSI-X is enabled) whenever requested by your application logic.

A simple handshake is required. The MSI-X interface is used only in the DM core in EP mode.

The following MSI interface signals are also used for MSI-X, depending on whether MSI or MSI-X is enabled: ven_msi_req, ven_msi_func_num, ven_msi_tc, and ven_msi_grant.

The address and data fields for MSI-X packets are defined in an MSI-X table located in the Endpoint's application logic, as shown in the figure below. Each entry in the table corresponds to an MSI-X packet that the application may send.

Each MSI-X table entry also includes a mask bit for that interrupt. An additional table, the pending bit array (PBA), includes a pending bit for each MSI-X table entry.

Host software sets and clears mask bits. If an interrupt's mask bit is set when the application wishes to send that MSI-X packet, the application must, instead, set the corresponding entry in the PBA.

Application logic monitors pending bits. If the corresponding mask bit is cleared, then the application sends the corresponding MSI-X packet, and clears the pending bit.

In addition, the MSI-X configuration registers in the Core include:

- Pointers to the function's MSI-X table and PBA
- Length of the function's MSI-X table
- An enable bit for MSI-X for that function
- A global mask for MSI-X

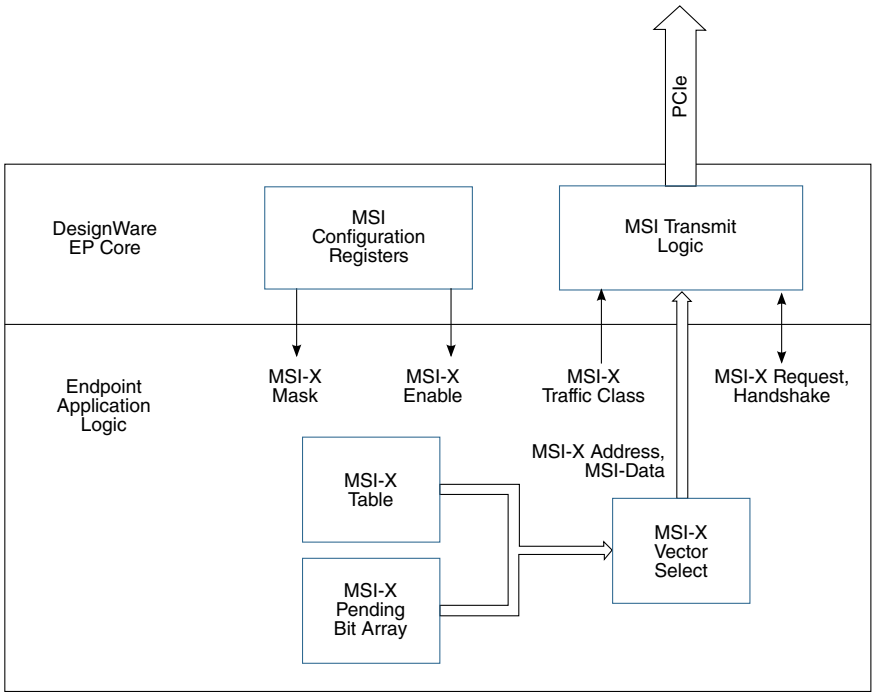


Figure 48-21. MSI-X Message Passing

Large MSI-X Tables

The MSI-X specification is written such that large MSI-X tables and PBAs may be implemented in compiled memories. However, if the host clears the function's global MSI-X mask, a search of the entire 2048 entry PBA might be necessary.

Also, in a large memory array like this, there would be no ìmemoryî of the order in which the pending bits were set. This is not an MSI-X requirement, but might be required by some systems.

48.3.7.2 Interrupts (EP Mode)

In EP mode, the core supports the Legacy PCI INTx compatible interrupt emulation mechanism, the Message Signaled Interrupts (MSI).

A function can use the Legacy PCI INTx mechanism. Different functions of the same device can use different mechanisms. For example, function 0 can use INTA while function 1 uses MSI.

Legacy PCI INTx Support

The core generates two Messages, ASSERT_INTX and DEASSERT_INTX, in response to assertion and the deassertion of the sys_int input. The Interrupt Pin register for each function determines whether the function uses INTA, INTB, INTC, or INTD.

In a single-function configuration, the core uses only INTA. An input signal, sys_int, is provided for the application to notify the core that an interrupt message should be sent. The sys_int signal is level-sensitive on a per-function basis. The rising edge transition of this signal triggers an interrupt assert message. The falling edge transition of this signal triggers an interrupt deassert message.

Message Signaled Interrupt (MSI) Support

MSI support is required for PCI Express devices. MSI-capable devices deliver interrupts by performing Memory Write transactions. The MSI is requested by application logic through the MSI interface; the core then generates the corresponding Memory Write.

48.3.7.3 Interrupts (RC Mode)

In RC mode, the core accepts ASSERT_INTX and DEASSERT_INTX Messages from the downstream component and provides the decoded output signals on the SII (for example, radm_inta_asserted and radm_inta_deasserted).

For interrupts the downstream component transmits through the MSI-X mechanism, the core passes the received Memory Write to the application on the RTRGT1 interface.

PCI Express Hot-Plug Logic Interrupt and Wakeup

In RC mode, the Hot-Plug logic supports generation of Hot-Plug interrupts on the following Hot-Plug events:

- Power Fault Detected
- MRL Sensor Changed
- Presence Detect Changed
- Command Completed
- Attention Button Pressed
- Electromechanical Interlock Status Changed
- Data Link Layer State Changed

When MSI or MSI-X mode is enabled, the core notifies the RC application of Hot-Plug events using the hp_msi output. When INTx interrupt mode is enabled, the core notifies the application of Hot-Plug events using the hp_int output.

If PME is enabled, the The Hot-Plug logic generates a Hot-Plug wake-up signal on hp_pme, triggered by the above Hot-Plug events. The RC Core does not check if the PM state is D1, D2, OR $D3_{HOT}$. It is up to the application to check the value on pm_dstate to make sure the device is in D1, D2, OR $D3_{HOT}$ upon receiving of hp_pme notification.

48.3.7.4 MSI Generation in the AXI Bridge

The standard AXI bridge simply sends MSI requests in the same manner as a memory write.

The method to send a MSI request is to have theAXI bridge send MSI requests in the same manner as a regular memory write. It is the application's responsibility to form the MSI request based on the native PCIe core's configuration. The native PCIe core CDM block contains the MSI address, enable, etc.

The application must obtain the MSI information (cfg_msi_*) from reading the MSI capability registers inthe CDM through the DBI interface, and then form the MSI request to present onto the AXI bridge slave interface.

This method should be used by an application that wants to preserve the order of Posted transfers (MemWr) requested before an MSI request.

The AXI bridge simply sends MSI requests in the same manner as a memory write. To send an MSI to the PCIe link from the AXI bridge, the application presents the MSI address and data onto the AXI slave write channel.

48.3.7.5 MSI Reception in the AHB/AXI Bridge

The standard AHB/AXI bridge receives MSI requests in the same manner as a memory write.

It cannot and does not distinguish between MSI and memory requests. The termination of an MSI request (in RC mode) must be done by the application or by using the optional MSI Controller described next.

48.3.7.5.1 AHB/AXI MSI Controller (Optional in RC mode)

The bridge provides an optional programmable MSI controller to detect and terminate inbound MSI requests in the bridge for RC and DM (RC mode) products. It is enabled by setting the CX_MSI_CTRL_ENABLE) configuration parameter.

Rather than propagating MSI MWr TLPs onto the AHB/AXI bus via the Master interface; the MSI packets are captured and terminated in the AHB/AXI Bridge and an interrupt is signaled.

The MSI Controller is programmed with an address that will be used as the system MSI address. If an inbound (received) MWr request is passed to the AHB/AXI Bridge and matches the specified MSI address as well as the conditions specified for an MSI Memory Write request, then an MSI interrupt is detected. When this Memory Write Request is about to be driven onto the AHB/AXI bridge Master Interface - it is quashed and never appears on the AHB/AXI bus.

The MSI Controller decodes the MSI MWr data payload to determine which End Point device (EP) sent the MSI and which interrupt vector it corresponds to. When a valid interrupt has been decoded, the msi_ctrl_int output is asserted. This output remains asserted when any MSI interrupt is pending. It is only deasserted when there is no MSI interrupt pending.

Features:

- MSI Interrupt Controller only enabled in RC core and DM core in RC mode based on the
- device_type input to DM. It is inactive in EP mode.
- The MSI Interrupt controller will provide support for up to eight EPs. Each supported EP will have a set of Interrupt Enable, Interrupt Mask and Interrupt Status registers.
- A maximum of 32 interrupts are supported per EP.

MSI Request Detection Criteria:

An MSI Interrupt Request is defined to occur when a Memory Write TLP that satisfies the following conditions is received by the core:

- Header Attributes bits are zero. No Snoop (NS) and Relaxed Ordering (RO) must be zero.
- Length field is 0x01 to indicate payload of one DWORD.
- First Byte Enable must be such that it is enabling the first 2 bytes (16-bits) of the payload.
- Last Byte Enable is 4'b0000.
- TLP address corresponds to system's chosen MSI Address as programmed in the MSI Controller Address Register. This register is not the MSI Lower 32 Bits Address Register which is part of the PCI Express MSI Capability Register structure.

In addition to the conditions outlined above, the Memory Write Request must also pass the receive filtering rules as outlined in Receive Filtering to be recognized as a valid MSI Interrupt Request. For example, Poisoned Bit not set in TLP header, ECRC check passed.

48.3.7.5.2 Programming and Usage Model

- The host CPU configures MSI capabilities of all Endpoints (EP) via the local DBI bus (or[1] via Config requests from the remote link partner).
- The MSI data register (MSI Data Register which is part of the PCI Express MSI Capability Register structure) of each EP is programmed as follows to allow the MSI Interrupt Controller to decode the interrupt source.

Table 48-19. MSI Data Register Programming for Use with AHB/AXI MSI Interrupt Controller

15:8	7:5	4:0
Not Used	EP Number •Allows each EP to be identified within the system, •For example, EP#5 is programmed with 3'b101.	Interrupt Vector Number •Identifies the interrupt source within each EP. •Programmed to 5'b00000. •Set by MSI generation logic to identify each interrupt source in real time. •Supports up to 32 Vectors.

- The MSI address register (MSI Lower 32 Bits Address Register which is part of the PCI Express MSI Capability Register structure) of each EP is programmed with the same message address.
- The host CPU configures the MSI Interrupt Controller via the local DBI bus (or via Config requests from the remote link partner).
- The common MSI Address that was used for the EPs is programmed into MSI Controller Address Register in the Interrupt Controller (MSI Controller Address Register).
- The Host CPU reads the MSI capabilities of each EP to determine the number of vectors enabled in each EP and uses this information to program the Interrupt Enable registers in the MSI Interrupt Controller. The Interrupt Enable register allows up to 32 MSI Interrupt Vectors to be enabled within the MSI Interrupt Controller for a given EP. It is the responsibility of the host CPU to read the contents of the "Multiple Message Enable" field in an EP's MSI capability structure and program that EP's Interrupt Enable register in the Interrupt Controller appropriately. For example, if Multiple Message Enable is 3'b100 for Endpoint N, which corresponds to 16 enabled Interrupt Vectors, then Interrupt Enable Register N in the Interrupt Controller should be programmed with 0x0000FFFF by the host CPU.

The MSI Interrupt Controller in your core is active and terminates all received MSI MemWr unless you deactivate it.

NOTE

The MSI Interrupt Controller is deactivated when:

- The core is not in RC mode (applies to DM product only). That is, device_type[3:0] is not 4'b0100.
- All of the eight Interrupt Enable Registers have a value of 0x0.

PROCESSING OF DETECTED INTERRUPTS

The MSI Controller decodes the MSI MWr data payload to determine which End Point device (EP) sent the MSI and which interrupt vector it corresponds to.

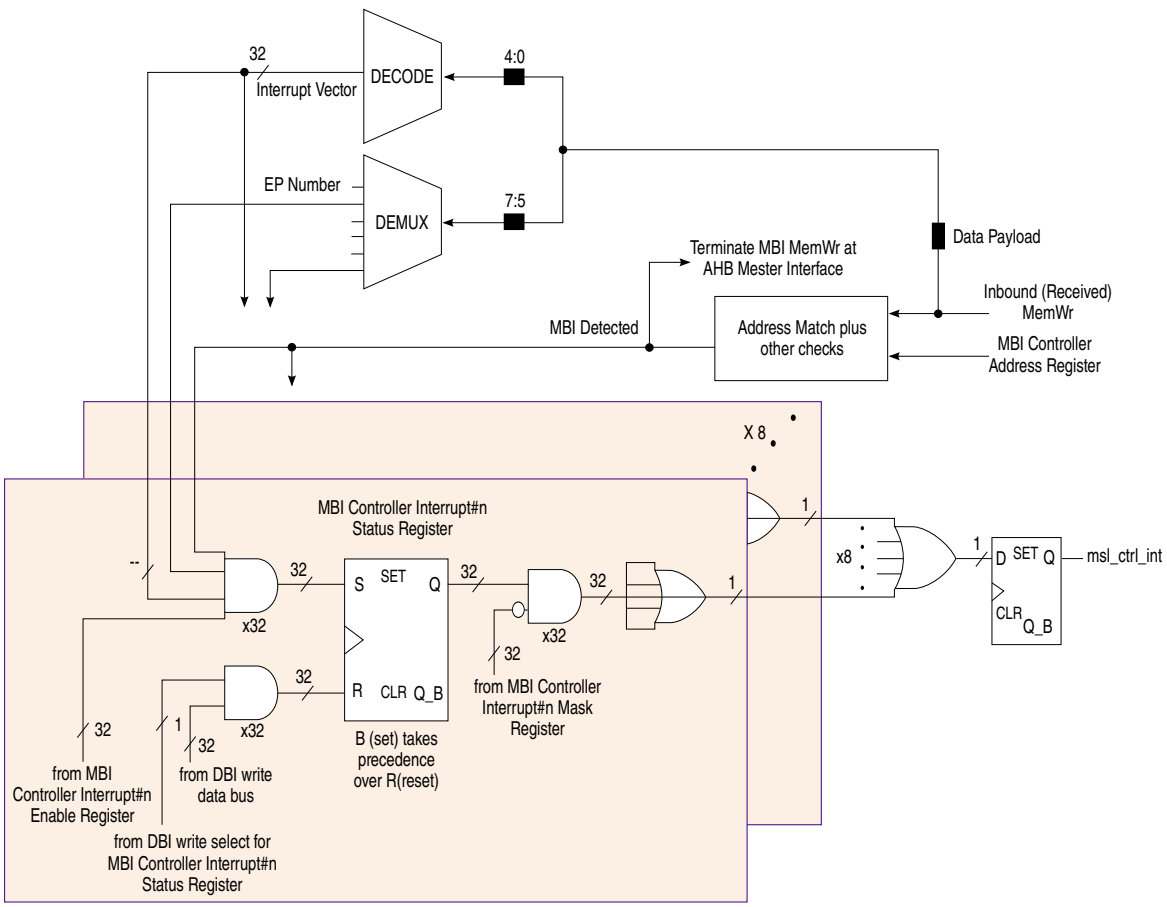


Figure 48-22. Architectural Representation of how the MSI Controller processes detected interrupts.

If the decoded Interrupt Vector is enabled and not masked then the corresponding bit is set in the Interrupt Status register (MSI Controller Interrupt#0 Status Register) and the top-level core output msi_ctrl_int is asserted. This signal remains asserted until the host CPU clears the status bit by writing a 1'b1 to the status bit. (Writing a 1'b0 has no effect). If any status bit remains set then msi_ctrl_int remains asserted. The Interrupt Status register provides a status bit for up to 32 Interrupt Vectors per Endpoint.

If the decoded Interrupt Vector is enabled but is masked then the corresponding bit is set in Interrupt Status register but the top-level core output msi_ctrl_int is not asserted.

If an MSI Interrupt Vector is received from an Endpoint but that Vector has not been enabled in the corresponding Interrupt Enable register (MSI Controller Interrupt#0 Enable Register) then no bit will get set in the Interrupt Status Register and msi_ctrl_int will not be asserted.

In addition, if no interrupts have been enabled in any of the eight Interrupt Enable Registers, then all MSI detection logic is disabled and valid MSI MWr request TLPs are not terminated in the bridge and are passed by the AHB/AXI master interface to the AHB/AXI bus.

The Interrupt Mask register (MSI Controller Interrupt#0 Mask Register) allows the Host to mask a given MSI Interrupt Vector. If a MSI Interrupt Vector is received for a masked Interrupt Vector, the corresponding bit in the Interrupt Status register will get set but msi_ctr_intl will not be asserted as the Interrupt Vector is masked. Note: This masking is local to the MSI Interrupt Controller and is not part of Per Vector Masking (PVM) in any of the downstream Endpoints.

The contents of the Interrupt Mask and Interrupt Status registers are used to drive the msi_ctrl_int output. If any status bit is set and the interrupt vector is not masked, then msi_ctrl_int is asserted HIGH.As long as any Interrupt Status bit is set and not masked, msi_ctrl_int will remain asserted.

ALTERNATIVE DATA REGISTER PROGRAMMING

The programming of the data register (MSI Data Registerwhich is part of the PCI Express MSI Capability Register structure) assumes each Endpoint enables 32 Vectors. If one or more Endpoints only support 16 or less Interrupt Vectors then it is possible to share a single set of Interrupt Enable, Interrupt Mask and Interrupt Status registers among those Endpoints.

For example, assume that Endpoint #1 enables 16 Interrupt Vectors and Endpoint #2 also enables 16 Interrupt Vectors. If the MSI Data register for Endpoint #1 is programmed to 0x20 and the MSI Data Register for Endpoint#2 is programmed to 0x30, then both Endpoint's Interrupt Status will be contained in Interrupt Status Register#1 with Endpoint#1's Vectors in the lower 16 bits and Endpoint#2's Vectors in the upper 16 bits. Manipulating the MSI Data Register format in this manner allows the Interrupt Controller to support more than 8 Endpoints using the 8 Interrupt Status registers provided.

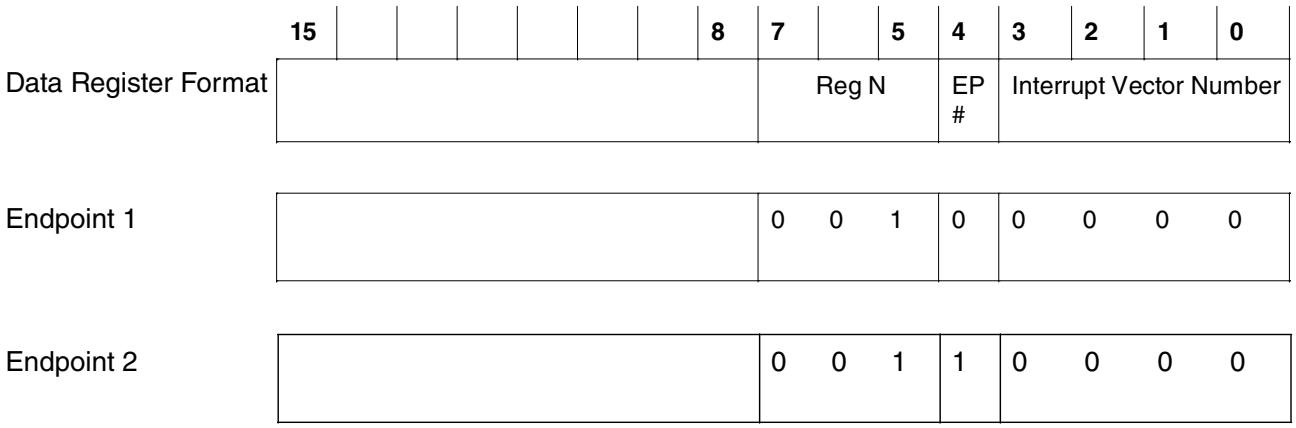


Figure 48-23. Alternative MSI Data Register Format

48.3.8 Flow Control

PCI Express implements a differentiated, credit-based Flow Control system to prevent overflows at the receiver.

In contrast to the simple XON/XOFF type flow control of the Ethernet protocol, the PCI Express Flow Control system requires that the consumer of data advertise the available buffer space for each type and priority of traffic. The Flow Control mechanism is divided into two phases: the initialization phase and the update phase. The core automatically performs both of these phases with minimal support required from the application.

The Flow Control for VC0 must be initialized following Link initialization, but prior to sending normal traffic. This initialization is performed by the Flow Control Initialization state machine. The initialization process involves exchanging information with the Link partner about the size of the receiver's buffers for each type of packet data: Posted, Non-Posted, and Completion. Header and payload buffers for each of these types are tracked and reported independently. Flow Control must be initialized for Virtual Channel 0 (VC0) before the Data Link Layer's link state moves into the DL_ACTIVE state, and normal traffic can begin flowing. Additional VCs (if any) are initialized following this, intermingled with the regular traffic already flowing on VC0. Initialization of other VCs begins when the VCs are enabled. The VC0 traffic has priority over non-VC0 flow control initialization.

The core provides a configurable solution to choose the number of credits to advertise per type and per VC. It can be configured to support multiple VCs as well as infinite credits. The core performs all required Flow Control protocol handshakes. The core currently provides a solution in which the application does not have to deal with Flow Control

updates and checks. By default, the RADM is responsible for returning Flow Control credits as data is read out of the RADM queuing structure. Additionally, the core provides optional signals to enable the application to handle Flow Control returns in an application-specific manner

The core does not return Flow Control credits for packets that have Data Link Layer errors.

48.3.9 Address Translation

This is a local address translation implementation and is not related to the PCI-SIG ATS specification support.

If address translation support is enabled for the core, it can use an address translation unit (ATU) to replace the TLP address and TLP header fields in the current TLP request header.

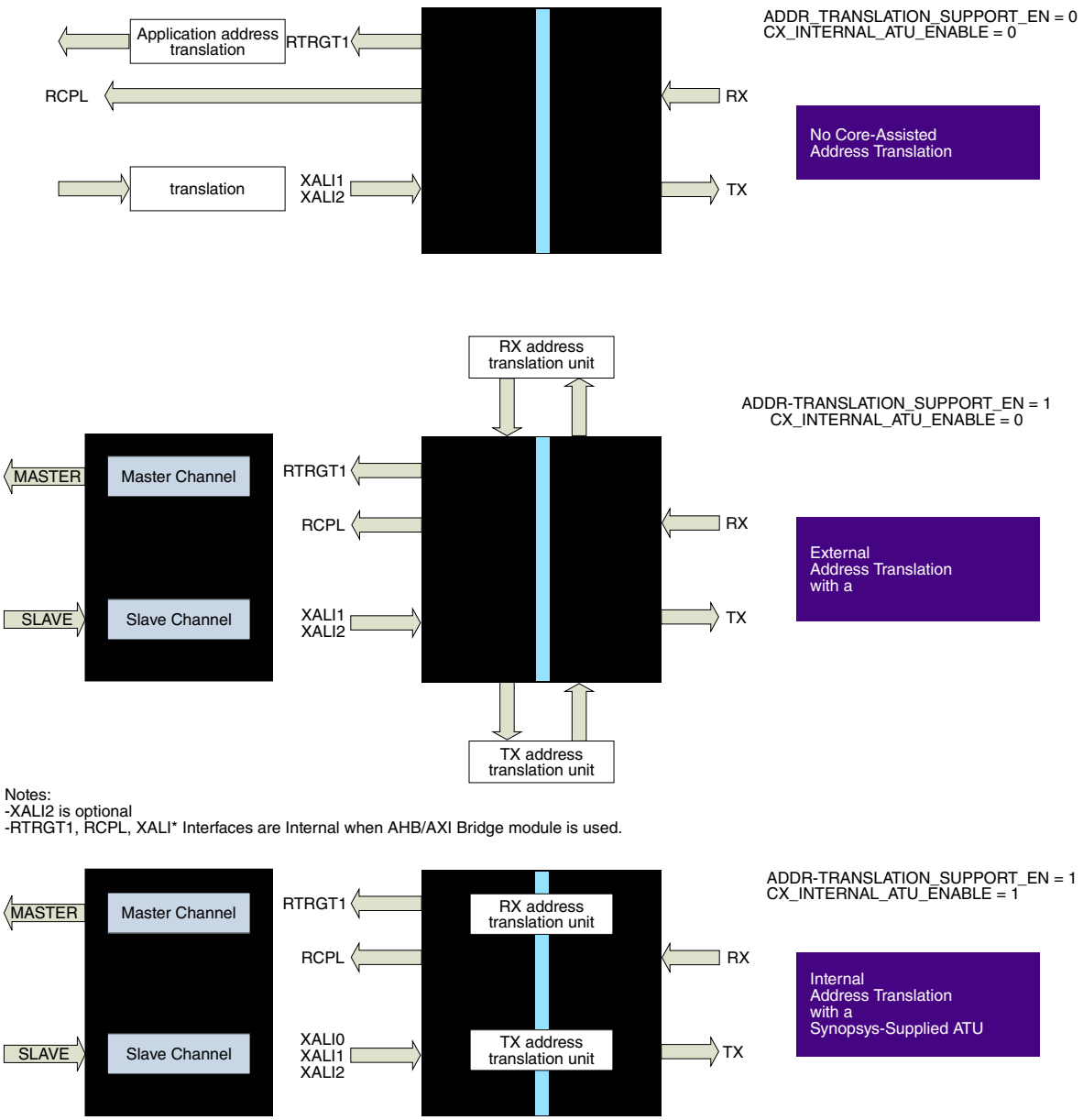


Figure 48-24. Address Translation options

Address translation is used for mapping different address ranges to different memory spaces supported by the application. A typical example will map the AMBA memory space to PCI memory space when the application has the PCIe-to-AMBA bridge in place. TYPE translation is also supported.

Without address translation, the application address is passed to/from the PCIe TLPs directly through AXI bridge interfaces.

The PCIe core supportsaddress translation.

1. Internal Address Translation (iATU) instantiates an internal ATU (iATU) inside the PCIe core. It can be configured (by software) to implement a customer-defined address translation scheme without the need for additional hardware from the customer. To enable this option:

Table 48-20. Configuration Parameters Relevant to Address Translation

Parameter Name	Parameter Label	Value Range	
CX_ATU_NUM_OUTBOUND_REGIONS	Number of Outbound Address Translation Regions	4,	
CX_ATU_NUM_INBOUND_REGIONS	Number of Inbound Address Translation Regions	4,1	
CX_ATU_MIN_REGION_SIZE	Minimum Size of Address Translation Region	64 kB	

NOTE

The easiest and recommended method of implementing address translation is to use the Internal Address Translation Unit (iATU).

48.3.9.1 Internal Address Translation (iATU)

An internal ATU (iATU) is instantiated inside the PCIe core.

It can be configured (by software) to implement a customer-defined address (and TYPE/ FORMAT) translation scheme without the need for additional external hardware.

48.3.9.1.1 Outbound (TX) Features

- Address Match Mode operation for MEM/IO/CFG/MSG TLPs. No address translation for CPL.
- Supports TYPE translation via TLP TYPE header field replacement for MEM[1] types to MSG/CFG types.
- This includes translation from Posted to Non-Posted (for example, MWr to CfgWr0).
- No TYPE translation from CPL TLPs.
- Programmable TLP header per region for the following fields for TLP field *replacement*.
- TYPE / TD / TC / AT / ATTR / MSG Code
- Function Number (Physical and Virtual).
- 4 Address Regions based on programmable registers for location and size.
- Programmable enable/disable per region.

- Automatic format (FMT) field translation between 3 DW and 4 DW for 64-bit addresses.
- Invert Address Matching Mode to translate accesses outside of a successful address match.
- ECAM Configuration Shift Mode to allow a 256 MB CFG1 space to be located anywhere in the 64-bit address space.
- Can be used to replace usage of misc sideband AXI slave bus signals.
- Supports regions from 64kB to 4 GB in size.
-

48.3.9.1.2 Inbound (RX) Features

- Address Match Mode operation for MEM/IO/CFG/MSG TLPs. No address translation for CPL. Selectable BAR Match Mode operation for IO/MEM TLPs.
- TLPs destined for RTRGT0 (internal CDM or ELBI) will not be translated.
- TLPs that are not error-free (ECRC, malformed and so on) will not be translated.
- Programmable TLP header per region for the following fields for *matching*.
- TYPE / TD / TC / AT / ATTR / MSG Code
- Function Number (Physical and Virtual).
- Up to 4 Address Regions based on programmable registers for location and size.
- Programmable enable/disable per region.
- Automatic format (FMT) field translation between 3 DW and 4 DW for 64-bit addresses.
- Invert Address Matching Mode to translate accesses outside of a successful address match.
- Configuration Shift Mode. Optimizes the memory footprint of CFG accesses destined for the AXI bridge interface in multi-function devices.
- Supports cores with and without the AHB/AXI Bridge module.
- Response Code defines the CPL completion status to return for accesses matching a region.
- Supports regions from64 kB to 4 GB in size.

48.3.9.1.3 Programming (iATU)

The iATU registers are in the PCIe cores' port logic register space (See PCIe CTRL Port Logic Memory Map/Register Definition). This may be accessed locally via the DBI interface or via PCIe Configuration accesses.

The following registers are used for programming the iATU.

Table 48-21. iATU Register Map

Byte Offset	Description

Table continues on the next page...

Table 48-21. iATU Register Map (continued)

+0x200	iATU Viewport Register
+0x204	iATU Region Control 1 Register
+0x208	iATU Region Control 2 Register
+0x20C	iATU Region Lower Base Address Register
+0x210	iATU Region Upper Base Address Register
+0x214	iATU Region Limit Address Register
+0x218	iATU Region Lower Target Address Register
+0x21C	iATU Region Upper Target Address Register

Full descriptions of each register are available at PCIe CTRL Port Logic Memory Map/ Register Definition.

48.3.9.2 Outbound iATU Operation

Information found here describes the processing of outbound requests by the iATU.

The topics for this section are:

- RID BDF Number Replacement
- iATU Outbound MSG Handling
- CFG Handling
- CFG Shift Feature
- FMT Translation
- Invert Feature
- No Address Match Result
- Writing to a MRdLk Region
- Programming Example

OVERVIEW (ADDRESS MATCH MODE)

The address field of each request MEM/IO TLP is checked to see if it falls into any of the enabled[1] address regions defined by the 'Start' and 'End' addresses. If an address match is found, then the TLP address field is modified as follows:

Address = Address - Base Address + Target Address

and the TYPE, TD, TC, AT and ATTR TLP header fields are replaced with the corresponding fields in iATU Control 1 Register.

If the application (AXI) address field matches more than one of the CX_ATU_NUM_OUTBOUND_REGIONS address regions, then the first (lowest of the numbers from 0 to CX_ATU_NUM_OUTBOUND_REGIONS-1) enabled region to be matched is used.

If there is no address match, then the address is untranslated. In the outbound direction (only), the TLP header information (for fields that are programmable) will come from the relevant fields on the AXI Slave Interface sideband busses (all fields will be 0).Since all misc_info are tied to 0, this is not recommended.

This operational mode (called Address Match Mode) is always used for outbound translation.

1. If the 'Region Enable' bit of the 'Region Control 2 Register' is '0', then that region is not used for address matching.

When the PCIe core is operating with 32-bit addresses, the operation is defined as in the following figure.

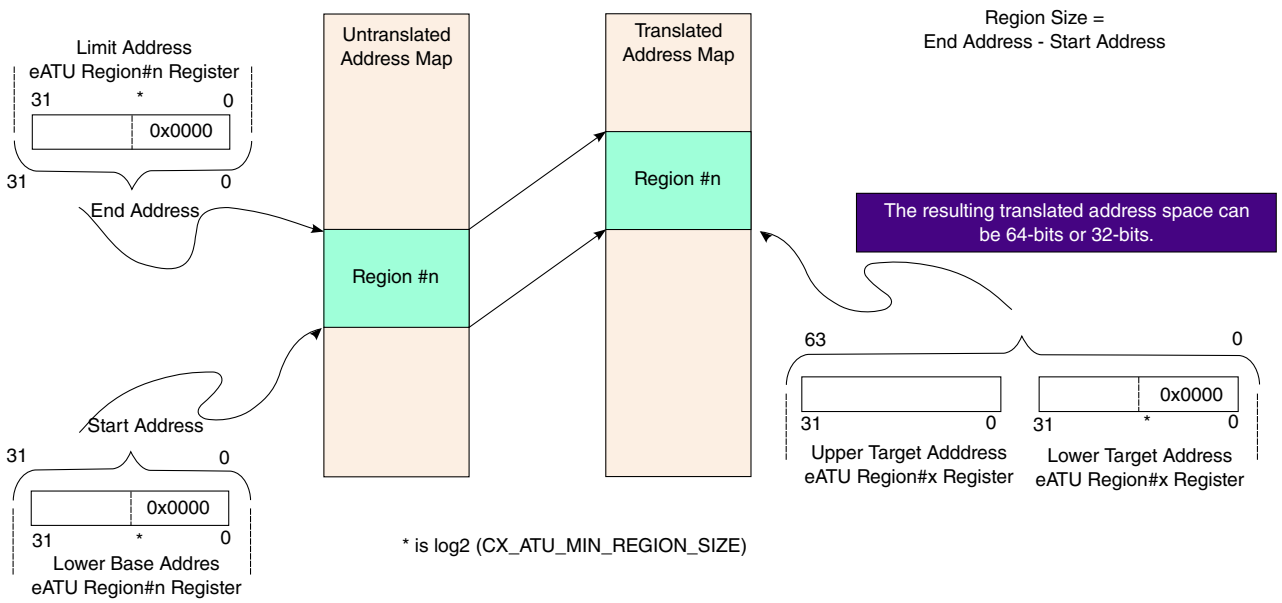


Figure 48-25. iATU Address Region Mapping: Outbound and Inbound (Address Match mode): 32-bit Address

The upper 32 bits of the Target Address Register will always form the upper 32 bits of the translated address because:

- The maximum region size is 4 GB.
- A region may not cross a 4 GB boundary.

The CX_ATU_MIN_REGION_SIZE (64 kB) specifies the minimum size of an address translation region. For example, if set to 64 kB; the lower 16 bits of the Base, Limit and Target registers are zero and all address regions are aligned on 64 kB boundaries. More precisely, the lower log2(CX_ATU_MIN_REGION_SIZE) bits are zero.

RID BDF NUMBER REPLACEMENT

When there is a successful address match on an outbound TLP, then the function number used in generating the 'Function' part of the Requester ID (RID[1]) field of the TLP is taken from the 3-bit 'Function Number' field of the iATU Control 1 Register. The value in this field must be 0x0 unless MultiFunction operation in the core is enabled (CX_NFUNC > 1).

iATU OUTBOUND MSG HANDLING

The iATU supports TYPE translation/conversion of MEM[4] TLP's to Msg/MsgD TLPs. This supports applications that are unable to generate Msg/MsgD type TLPs natively.

When there is a successful address match on an outbound MEM TLP, and the translated TLP TYPE field is 'Message' (that is, 'TYPE' field of the iATU Control 1 Register is 10xxx); then the Message Code field of the TLP is set to the value in the 'Message Code' field of the iATU Control 2 Register.

A MWr with an 'effective length of 0' (see [5]) is converted to Msg and all other MWr TLPs are converted to MsgD.

NOTE

For more information on generating Messages, see Message Generation.

For a proper understanding of Messages, you should be familiar with the PCI Express Base Specification.

MSG translation is possible with the iATU. For MSG transactions created directly by the application (as opposed to, by the iATU) you must ensure that the 3rd and 4th DWORD (In the PCI Express Base Specification) does not match any programmed iATU address region or else unintentional translation (TYPE) could occur.

RID uses the 8-bit.5-bit.3-bit PCI Bus.Device.Function (BDF) format.

The maximum size of this field is 8 bits, but the actual size depends on the number of Virtual functions (VFs) used as denoted by 2^NVF_WD Or IO.

For AXI, this takes the First and Last byte enables into account. For AXI this is through slv_wstrb. If you are just translating the address of MSG TLPs, then client0_tlp_byte_en is used to provide the Message Code.

CFG HANDLING

Outbound CFG transactions (formed by translation of IO/MEM TLPs from the application) can exist anywhere in address space -because the Routing ID or BDF, is created by the iATU from bits [31:16] of the untranslated address - and this BDF changes according on the PCIe bus topology.

In the normal untranslated transmission of CFG transactions, the PCIe core derives the CFG Bus.Device.Function (BDF) information from the address on the AXI Slave Interface address) as follows.

Table 48-22. Normal PCIe Core Outbound Derivation of BDF from XALI* or AHB/AXI Slave Interfaces

XAXI Slave Interface Bits	PCIe CFG Header Field
31:24	Bus Number
23:19	Device Number
18:16	Function Number
11:8	Extended Register Number
7:2	Register Number

The iATU supports translation (of address or type) of IO/MEM TLP's to CFG TLPs. This supports applications that are unable to generate CFG type TLPs natively.

The 16-bit Bus.Device.Function (BDF) is derived from bits [31:16] of the iATU Lower Target Address Register.

CFG translation is possible with the iATU. For CFG transactions created directly by the application (as opposed to, by the iATU) you must ensure that the Bus.Device.Function field does not match any programmed iATU address region or else unintentional translation (TYPE) could occur.

CFG SHIFT FEATURE

A expander feature (CFG Shift Feature) can be enabled by setting the 'CFG Shift' bit of the iATU Control 2 Register.

This shifts/maps the BDF - bits [27:12] of the Target Address up to bits [31:16] of the translated address. This allows all outgoing IO/MEM TLPs (that have been translated to CFG) to be mapped into any 256 MB region of the PCIe address space.

This scheme also supports the Enhanced Configuration Address Mapping (ECAM) mechanism from the *PCI Express, Revision 3.0 Base Specification.* ECAM supports the mapping (via MEM to CFG TYPE translation) from memory address space to PCI Express Configuration Space address. ECAM maps bits [27:12] of the untranslated MEM TLP to become the BDF of the resulting CFG TLP.

Table 48-23. ECAM Scheme from PCIe Specification using an 8-bit BDF Bus Number

Memory Address Bits	PCIe Configuration Space
27:20	Bus Number
19:15	Device Number
14:12	Function Number
11:8	Extended Register Number
7:2	Register Number

FMT TRANSLATION

The iATU automatically sets the TLP format (FMT) field for 3DW when it detects all zeroes in the upper 32- bits of the *translated* address. Otherwise, it sets it to 4DW when it detects a 64-bit address (that is, when there is a '1' in the upper 32-bits of the translated address). If the original address and the translated address are of different format, the iATU ensures that the TLP header size matches the translated address format.

INVERT FEATURE

Normally, an address match on an outbound TLP occurs, occurs when the untranslated address is in the region bounded by the Base Address and Limit Address.

When the Invert feature is activated, an address match occurs when the untranslated address is NOT in the region bounded by the Base Address and Limit Address.

This feature is activated by setting the 'Invert' field of the iATU Control 2 Register.

NO ADDRESS MATCH RESULT

When there is no address match, then the address is untranslated but

the TLP header information will come from the relevant fields on the AHB/AXI Slave Interface sideband busses (all fields will be 0).

WRITING TO A MRDLK REGION

When there is a successful address match for an outbound WRITE, and the TYPE header field - as replaced with the TPYE field in iATU Control 1 Register - is MRdLk, then the TYPE header field will be set to MEM (that is 00000b).

PROGRAMMING EXAMPLE

See #d13869e5a1310 for details on the programming registers.

NOTE

Define Outbound Region 1 as:

NOTE

IO region from 0x80000000_d000000 - 0x80000000_d000ffff (64k)

NOTE

mapped to 0x00010000 in PCIe IO space.

1. Setup the Viewport Register

Write 0x00000001 to Address { 0x700 + 0x200 } to set outbound region 1 as the current region

2. Setup the Region Base and Limit Address Registers

Write 0xd0000000 to Address {0x700 + 0x20C} to set the Lower Base Address. Write 0x80000000 to Address {0x700 + 0x210} to set the Upper Base Address. Write 0xd000ffff to Address {0x700 + 0x214} to set the Limit Address

3. Setup the Target Address Registers

Write 0x00010000 to Address {0x700 + 0x218} to set the Lower Target Address

Write 0x00000000 to Address {0x700 + 0x21C} to set the Upper Target Address

4. Configure the region via the Region Control 1 Register

Write 0x00000002 to Address {0x700 + 0x204} to define the type of the region to be IO.

5. Enable the region

Write 0x80000000 to Address {0x700 + 0x208} to enable the region.

48.3.9.2.1 Inbound iATU Operation

This section describe the processing of inbound requests by the iATU. The topics for this section are:

- Overview
- IO/MEM Match Modes
- CFG Handling
- Optional Matching Fields
- Response Code Feature
- iATU Inbound MSG Handling
- Fuzzy Type Match Mode
- FMT Translation
- Invert Feature
- Programming Examples

48.3.9.3 Overview (iATU)

The main difference between Inbound and Outbound iATU operation is that the TLP TYPE is never changed in the Inbound direction.

Instead, the TYPE field is used for more precise matching. Other fields may also be optionally used to further refine the matching process.

Another difference is that for MEM/IO TLPs, you can select between Address matching (as used in Outbound Operation) or BAR matching. Normally an End Point (EP) will use BAR match mode and a Root Complex (RC) will use Address mode as an RC normally has no BAR's implemented.

Lastly, for CFG0 TLPs, you can select between Routing ID matching or Accept mode.

If there is no match then the address is untranslated. In addition,

- TLPs destined for RTRGT0 (internal CDM or ELBI) will not be translated.
- TLPs that are not error-free (ECRC, malformed and so on) will not be translated.
- Address translation of all TLP types (MEM/IO/CFG/MSG) except CPL is supported in Address Match mode. In BAR Match mode only translation of IO/MEM is supported.

IO/MEM MATCH MODES

Inbound Address translation for IO/MEM TLPs will operate in one of two matching modes as determined by the 'Inbound Match Mode' field in the iATU Region Control 2 Register.

1. Address Match Mode

The operation is similar to Outbound iATU Operation.

The address field of each request TLP is checked to see if it falls into any of the enabled[1] address regions defined by the 'Start' and 'End' addresses. If an address match is found, then the TLP address field is modified as follows:

Address = Address - Base Address + Target Address

If the TLP address field matches more than one of the CX_ATU_NUM_INBOUND_REGIONS address regions, then the first (lowest of the numbers from 0 to CX_ATU_NUM_INBOUND_REGIONS-1) enabled region to be matched is used.

2. BAR Match Mode

Looking for an address match is a two-step process.

1. The address field of MEM/IO *(only)* request TLPs is checked by the standard internal *PCI Express BAR Matching Mechanism* to see if it falls into any address region defined by the enabled BAR Addresses and Masks.
2. If a matched BAR was found, then that matched BAR ID is compared by the iATU to the 'BAR Number' field in the iATU Region Control 2 Register for all enabled regions.

BAR Match Mode can only be used for MEM/IO transactions.

Address Match Mode should always be used to match MSG transactions as these will never generate a match against a BAR.

1. If the 'Region Enable' bit of the 'Region Control 2 Register' is '0', then that region is not used for address matching.

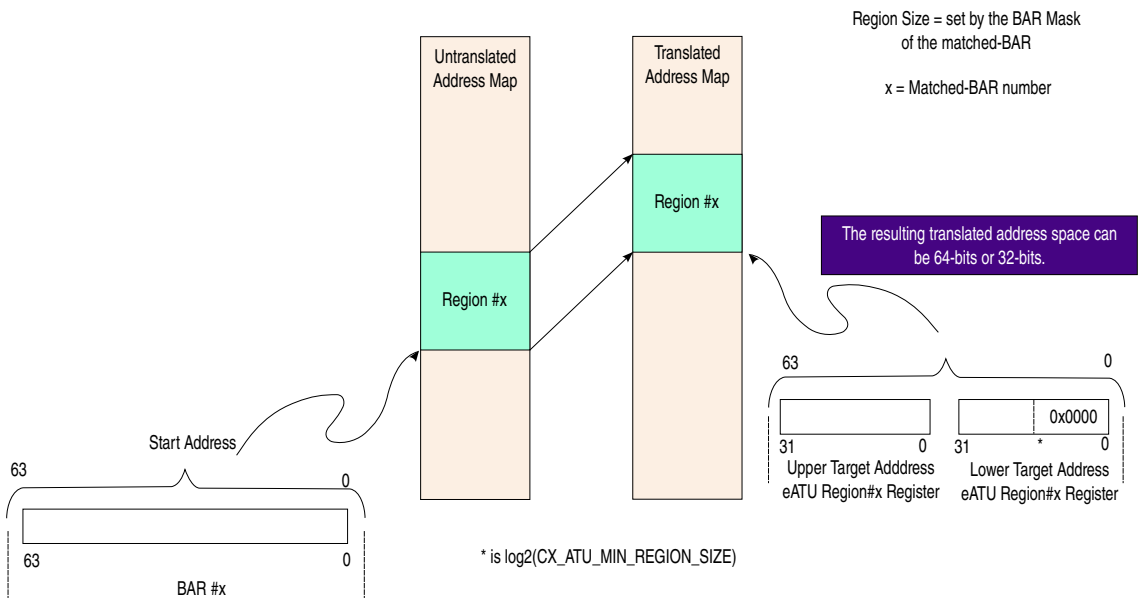


Figure 48-26. iATU Address Region Mapping: Inbound (BAR Match mode): 64-bit BAR

Normally an EP will use BAR match mode and an RC will use Address Match mode -as an RC normally has no BAR's implemented or at least must handle requests which do not match any of its BARs.

However, the user has the freedom to implement any mode in their device. For example, an EP device may use Address Match mode, but should be aware that if the address range does not match one of its BAR ranges in an EP, the device will reject the request with Unsupported Request (UR) completion status and no translation will occur.

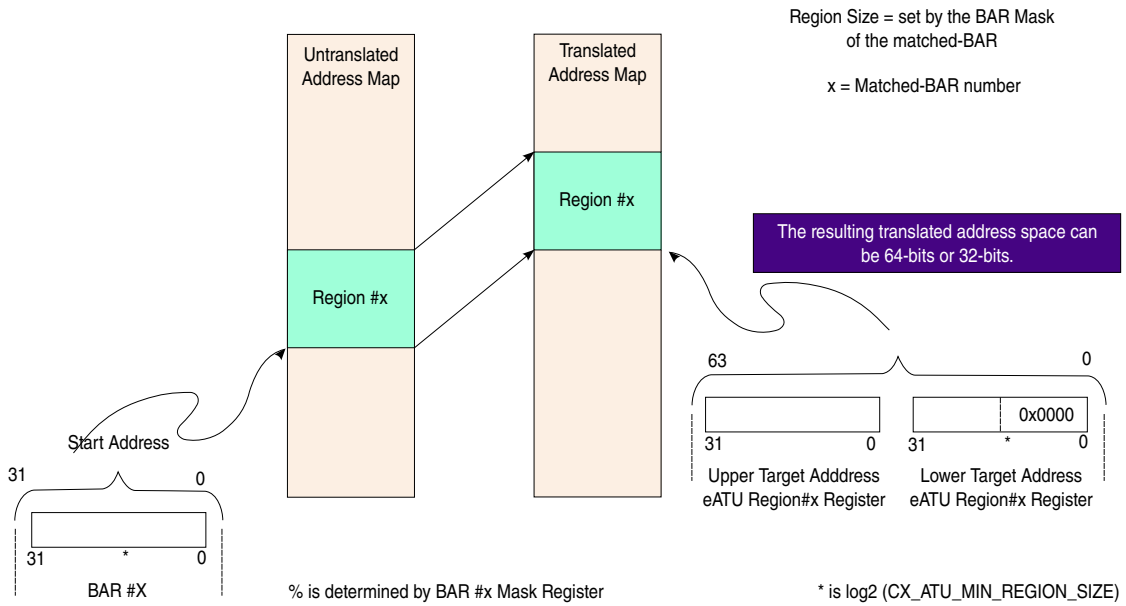


Figure 48-27. iATU Address Region Mapping: Inbound (BAR Match mode): 32-bit BAR

dress space

CFG HANDLING

CFG TLPs are normally routed to internal CDM. These will not be translated. Only CFG0 TLPs routed to AXI Bridge Master interface will be translated.

Inbound Address translation for CFG0 TLPs will operate in one of two matching modes as determined by the 'Inbound CFG0 Match Mode' field of the iATU Region Control 2 Register.

1. Routing ID Match Mode:

The operation is similar to outbound iATU operation. The Routing ID of the inbound CFG0 TLP must fall within the Base and Limit of the defined iATU region for matching to proceed. The iATU interprets the Routing ID (Bytes[1] 8 to 11 of TLP header) as an address. This corresponds to the upper 16 bits of the address in MEM/IO transactions.

2. Accept Mode:

CFG0 TLPs should always be accepted and processed even if the bus number does not match the current Bus number of the device. This mode follows that behavior. The Routing ID of received CFG0 TLPs will be ignored when determining a match.

CFG1 TRANSACTIONS

For CfgRd1/CfgWr1 transactions the Base and Limit Address could enclose the entire 32-bit 4G memory space with Routing ID forming the upper 16 bits. The Target Address maps these CFG transactions to anywhere in application address space.

CFG SHIFT FEATURE

Inbound CFG transactions (routed for AXI Bridge Master) can exist anywhere in address space -because the Routing ID or BDF, is processed by the PCIe core RADM filter (see Receive Application-Dependent Module (RADM)) as bits [31:16] of an address - and this BDF changes according on the PCIe bus topology.

A compressor feature (CFG Shift Feature) can be enabled by setting the CFG Shift bit of the iATU Region Control 2 Register. Bits [15:12] of the 3rd DWORD[1] of CFG TLPs are reserved. The compressor feature uses this to reduce the memory requirement.

This shifts/maps the BDF - bits [31:16] of 3rd header DWORD which would be matched against the Base and Limit Addresses - of the incoming CfgRd0/CfgWr0 down to bits [27:12] of the translated address.

OPTIONAL MATCHING FIELDS

In Address or BAR Match mode, a successful address/BAR match can be optionally gated by successful matching of the following programmable TLP header fields (per region):

- TYPE / TD / TC / AT / ATTR
- MSG Code (MSG TLP's only)
- Function Number (MEM, IO or CFG TLPs only)
- Virtual Function Number (MEM or IO TLPs only)

For each of the above fields in the iATU Region Control 1 Register (ATU Region Control 2 Register for MSG) there is an associated 'Match Enable' bit in the iATU Region Control 2 Register. Address translation will only proceed if all enabled field-matches are successful.

If SR-IOV is enabled (CX_SRIOV_ENABLE=1), and the Virtual Function Match Enable field of the iATU Region Control 2 Register is set, then the 'Function' is no longer the 3-bit 'Function' but the combined 8-bit 'Device' 'Function' parts. When SR-IOV is enabled, the Alternate RID Interpretation (ARI) RID scheme is used. This uses a 8-bit.0-bit.8-bit BDF format. where the device number is assumed to be zero.

RESPONSE CODE FEATURE

When the 'Response Code' field of the inbound iATU Region Control 2 Register is set to a value other than 00b, it will determine the Completion Status of the CPL TLP sent in response to a successfully matched Non Posted TLP. This can be set to Unsupported Request (UR) or Completer Abort (CA). When the error response field is set to 00b, then the normal RADM (see Receive Application-Dependent Module (RADM)) filter response for this TLP will be used.

IATU INBOUND MSG HANDLING

Inbound Message (Msg/MsgD) transactions can use one of two matching modes:

Address Match Mode: The 3rd and 4th header DWORDs are treated as an address and matched against the iATU Region Base and Limit Address registers. Furthermore, for Vendor Defined messages, this allows specific vendor defined messages to be filtered into memory at the Target Address. The Upper Base Address should be set to Bus.Device.Function (BDF) and Vendor ID. The Lower Base Address can be used as a filter for specific messages.

Vendor ID Match Mode: This mode is relevant for ID-routed Vendor Defined Messages. The iATU ignores the Routing ID (Bus, Device, Function) in bits [31:16] of the 3rd DWORD of the TLP header[1], but matches against the Vendor ID in bits [15:0] of the 3rd DWORD of the TLP header (bytes[1] 10 and 11). This allows Vendor defined messages to be filtered against specific Vendor IDs without needing to know the BDF number which may vary depending on the PCI topology.

Bits [15:0] of the Region Upper Base register should be programmed with the required Vendor ID as follows:

- Region Upper Base[15:8] = byte 10
- Region Upper Base[7:0] = byte 11

The lower Base and Limit Register should be programmed to translate TLPs based on vendor specific information in the 4th DWORD of the TLP header.

NOTE

For more information on generating Messages, see Message Generation. For a proper understanding of Messages you should be familiar with Message Request Rules of the PCI Express Base Specification.

FUZZY TYPE MATCH MODE

When enabled, the iATU relaxes the matching of the TLP TYPE field against the expected TYPE field so that

- CfgRd0 and CfgRd1 TLPs are seen as identical. Similarly with CfgWr0 and CfgWr1.
- MRd and MRdLk TLPs are seen as identical
- The Routing field of MsgD TLPs is ignored

For example, CFG0 in the TYPE field in the iATU Control 1 Register will match against an inbound CfgRd0, CfgRd1, CfgWr0 or CfgWr1 TLP.

To enable this feature, then set the 'Fuzzy Type Match Mode' bit of the iATU Region Control 2 Register.

FMT TRANSLATION

The iATU automatically sets the TLP format (FMT) field for 3DW when it detects all zeroes in the upper 32- bits of the *translated* address. Otherwise, it sets it to 4DW when it detects a 64-bit address (that is, when there is a '1' in the upper 32-bits of the translated address). If the original address and the translated address are of different format, the iATU ensures that the TLP header size matches the translated address format.

INVERT FEATURE

Normally, an address match on an outbound TLP occurs, occurs when the untranslated address is in the region bounded by the Base Address and Limit Address.

When the Invert feature is activated, an address match occurs when the untranslated address is NOT in the region bounded by the Base Address and Limit Address.

This feature is activated by setting the 'Invert' field of the iATU Region Control 2 Register.

PROGRAMMING EXAMPLES

Define Inbound Region 2 as: MEM region matching BAR4 (BAR Match mode) mapping to 0x8000000020000000 in the application memory space

1. Setup the Viewport Register
   - Write 0x80000002 to Address { 0x700 + 0x200 } to set inbound region 2as the current region
2. Setup the Target Address Registers
   - Write 0x20000000 to Address {0x700 + 0x218} to set the Lower Target Address
   - Write 0x80000000 to Address {0x700 + 0x21C} to set the Upper Target Address
3. Configure the region via the Region Control 1 Register
   - Write 0x00000000 to Address {0x700 + 0x204} to define the type of the region to be MEM.
4. Enable the region for BAR Match Mode
   - Write 0xC0000400 to Address {0x700 + 0x208} to enable the region for BAR match mode for BAR#4.

Define Inbound Region 0 as: MEM region matching TLPs with addresses in the range 0x00010000 - 0x0005ffff mapped to 0x1000000020000000 - 0x100000002004ffff in the application memory space

1. Setup the Viewport Register
   - Write 0x80000000 to Address { 0x700 + 0x200 } to set inbound region 0 as the current region
2. Setup the Region Base and Limit Address Registers

- Write 0x00010000 to Address {0x700 + 0x20C} to set the Lower Base Address.
  Write 0x00000000 to Address {0x700 + 0x210} to set the Upper Base Address.
  Write 0x0005ffff to Address {0x700 + 0x214} to set the Limit Address

3. Setup the Target Address Registers
   - Write 0x20000000 to Address {0x700 + 0x218} to set the Lower Target Address
   - Write 0x10000000 to Address {0x700 + 0x21C} to set the Upper Target Address
4. Configure the region via the Region Control 1 Register
   - Write 0x00000000 to Address {0x700 + 0x204} to define the type of the region to be MEM.
5. Enable the region
   - Write 0x80000000 to Address {0x700 + 0x208} to enable the region in address match mode.
   - EP: Defined MEM or IO regions must be inside an enabled BAR range.
   - RC: Defined MEM or IO regions must either match a BAR or be outside of the base and limit ranges defined for the port in the Type 1 configuration header.

48.3.10 Gen2 5.0 GT/s Operation

The PCIe Express core supports all of the non-optional Gen2 5.0 GT/s features defined in the *PCI Express 3.0 Specification.*.

48.3.10.1 Overview (Gen2 5.0 GT/s)

The DWC PCIe cores support achives the PCI Express 2.0 rate, by DYNAMIC

FREQUENCY

1. When supporting Gen2 DYNAMIC FREQUENCY, the core operates at either 125 MHz at the Gen1 rate. When operating at the Gen2 rate, the core's clock frequency is changed to 250 MHz

Software configuration of Gen2 5.0 GT/s operation is available through the Gen2 Control Register.

48.3.10.2 Speed Changing

If bit 17 ìDirected Speed Changeì of the Gen2 Control Register is set to '1', then the LTSSM will initiate a speed change after the link is initialized. The default value of this register is the '1' .

A PIPE signal, mac_phy_rate, is used to negotiate the Link data rate. The core drives mac_phy_rate to indicate the negotiated Link data rate. For mac_phy_rate, a value of zero indicates 2.5 Gbps, and a value of one indicates 5.0 Gbps. The PCIe core changes the rate signal and waits for a pulse on the phy_mac_phystatus signal to confirm that the PHY has accepted the requested rate.

The PCIe core uses core_clk to sample the phy_mac_phystatus signal during speed changes. The PHY generates phy_mac_phystatus based on pipe_clk. Therefore, the following restrictions are placed on DWC_pcie external logic when the PHY is Gen2 Dynamic Frequency and the PCIe core is Gen2 Dynamic Width (i.e. when core_clk is not equivalent to pipe_clk).

1. ensure that core_clk is toggling when the PHY returns phy_mac_phystatus for a speed change

OR

2. the external logic must hold phy_mac_phystaus (for the speed change) until core_clk toggles again

For other configurations, the core_clk is equivalent to the pipe_clk, and this restriction is not needed.

48.3.11 Power Management

An architectural overview of the Power Management Controller is given in "Power Management Controller (PMC)".

There are two types of power management operations:

- Software controlled PCI power management operations
- Active state power management operation (ASPM) for PCIe device only

The following figure shows the capable link state and its control conditions.

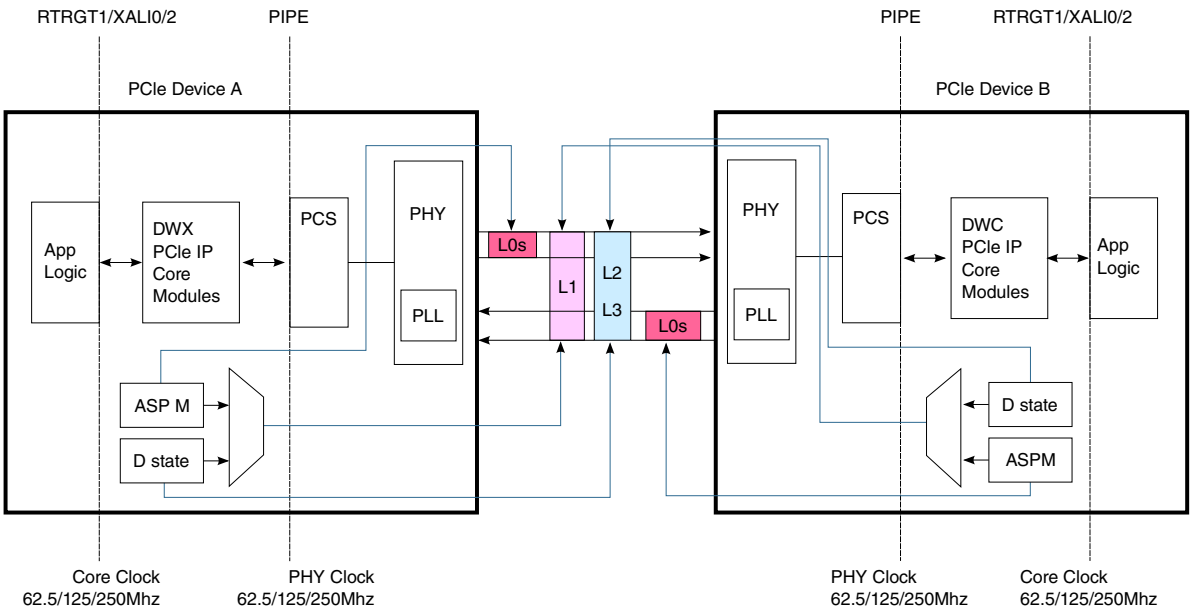


Figure 48-28. Power Management - Capable Link State and Control Conditions

The L0s link state is controlled by the ASPM L0s enter condition met state. The L1 link state is controlled either by the ASPM L1 enter condition met state, or by the D-state (D1, D2, or D3) of the PCIe device. The D- state of the PCIe device is programmable by software. The L2/L3 ready state is controlled by D-state and power turn-off event.

The power saving of links in lower power states is greater as the link state numbers get larger. The following figure shows the links states of PCIe devices and the relationships of power down states between link partners.

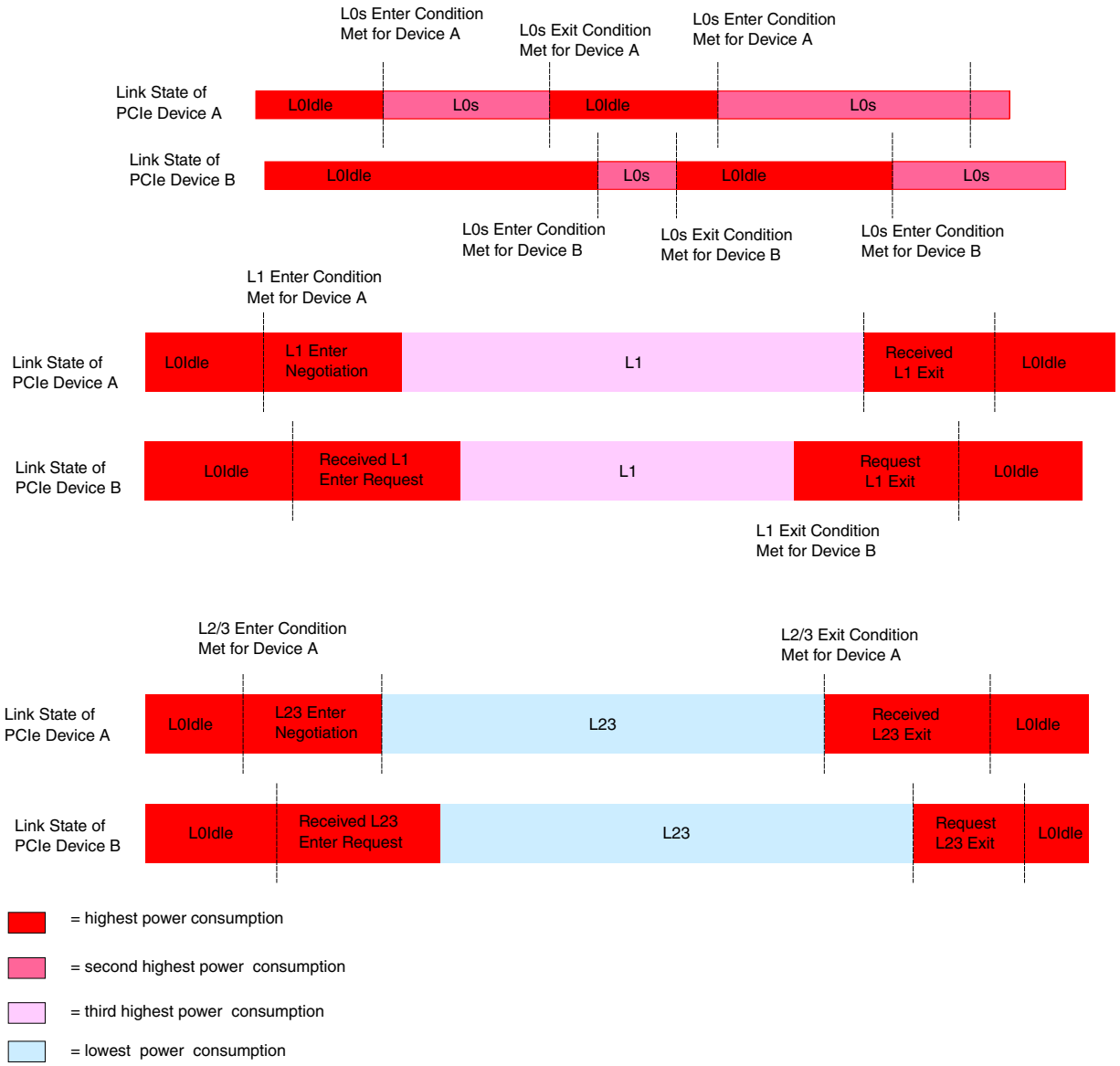


Figure 48-29. Relationships of Power Down States between Link Partners

48.3.11.1 L0s Power Down

L0s is a low power state enabled by Active State Power Management (ASPM). ASPM enabled devices can only control L0s entrance of the transmitter. The receiver L0s is controlled by the remote devices.

L0S ENTER CONDITIONS (ALL CONDITIONS MET)

- ASPM L0s is enabled.
- L0s enter conditions defined by *PCI Express Specification* for a duration of time and there is no higher stage of power down requested.
- The timeout value is controlled by the DEFAULT_L0S_ENTR_LATENCY parameter.

L0S EXIT CONDITIONS (ANY CONDITION MET)

- Any DLLP or TLP pending to be sent.
- L1 enter condition met.
- PCIe link partner request to enter into link recovery.

48.3.11.2 L1 Power Down

L1 is a power down state enabled either by ASPM or by the software controlled D1, D2 or D3 state (which is programmed by the system power management unit). L1 state is a bi-directional link power down state. Both link partners must negotiate to go to L1 state.

L1 ENTER CONDITIONS DUE TO ASPM (ALL CONDITIONS MET)

There are three scenarios that result in the L1 state:

- Scenario 1: L1 Idle Timeout From L0s

ASPM L1 and L0s are enabled.

Link state is in L0s for both transmitter and receiver of the link, and bit 30 of the Ack Frequency and L0-L1 ASPM Control Register is set to 0 (default setting) OR Link state is in L0s of transmitter and bit 30 of the Ack Frequency and L0-L1 ASPM Control Register is set to 1.

L1 enter conditions defined by PCIe spec for a duration of time and there is no higher stage of power down requested.

The timeout value is controlled by the DEFAULT_L1_ENTR_LATENCY parameter.

- Scenario 2: L1 Idle Timeout from L0

ASPM L1 is enabled and L0s is not enabled.

Link state is in L0.

L1 enter conditions defined by PCIe spec for duration of time, and there is no higher stage of power down requested.

The timeout value is controlled by the DEFAULT_L1_ENTR_LATENCY parameter.

- Scenario 3: Application Controlled

ASPM L1 is enabled.

Application request to enter L1 by asserting signal app_req_entr_l1

L1 enter conditions defined by PCIe spec is met.

NOTE

The app_req_entr_l1 is a pulse. The core latches the command and makes sure that L1 has been entered.

L1 ENTER CONDITIONS DUE TO D1/D2/D3 STATES (ALL CONDITIONS MET)

- All functions that are programmed to D1, D2 or D3 states.
- Always enter L1 when L2/L3 PM turn-off negotiation has not yet been done.

L1 EXIT CONDITIONS (ANY CONDITION MET)

- Software requests a higher stage of power down.
- Any DLLP or TLP pending to be sent.
- Application requesting exit of L1 by asserting signal app_req_exit_l1.
- Link partner requesting exit of L1.

Once L1 has exited, another L1 entry will not be initiated for 10us if the enter L1 condition is due to ASPM. If the enter L1 condition is due to lower power D-state, the core will enter L1 again after a wait time of cfg_cpl_sent_count cycles defined in PL register. This wait time to ensures the exit conditions have been served.

3.14.3 L2/L3 Power Down

The core has control over the L2 or L3 ready link state. After the L2/L3 ready is entered, the downstream device will begin preparation for the power and clock removal. After main power has been removed, the link will transition to L2 if Vaux is provided, or it will transition to L3 if no Vaux is provided. L2/L3 ready is a bi-directional link power down state.

L2/L3 ENTER CONDITIONS (ALL CONDITIONS MET)

- PME_Turn_Off/Pme_To_Ack handshake has been completed at any of D0,D1,D2,D3 states.
- Application is ready to be turned off by asserting signal app_ready_entr_l23.

L2/L3 EXIT CONDITIONS (ANY CONDITION MET)

- Device is programmed with capability to support PME and application requests wakeup by asserting the apps_pm_xmt_pme signal or by triggering a native hot-plug event when D-state is in D1, D2 or D3.
- Link partner requesting exit of L2/L3.

The core supports beacon signaling by asserting signal pm_phy_beacongen or wake when a wake-up event is initiated by a PCIe device.

48.3.12 Completion Timeout Ranges

Timeout ranges are supported as defined in the *PCI Express 3.0 Specification*.

The Device Capabilities 2 Register (Offset 24h) shows support for all ranges. The Device Control 2 Register (Offset 28h) will have a reset value equal to the default value in the spec: "0000b Default range: 50 us to 50 ms". If the default value is used then the timeout will be in "Range B: 0101b: 16ms to 55ms." This range was chosen for the default because the *PCI Express 3.0 Specification* states "It is strongly recommended that the Completion Timeout mechanism not expire in less than 10 ms." The following table illustrates the specification values versus the PCI Express core values for the ranges.

NOTE

As per the PCIe 2.1 spec, "This mechanism is intended to be activated only when there is no reasonable expectation that the Completion will be returned, and should never occur under normal operating conditions."

Table 48-24. PCIe Core Completion Timeout Ranges versus PCI Express Specification

Range	Encoding	Spec Minimum	Spec Maximum	PCIe Core Minimum	PCIe Core Maximum
Default	0000b	50µs	50ms	28ms	44ms
A	0001b	50µs	100µs	65µs	99µs
A	0010b	1ms	10ms	4.1ms	6.2ms
B	0101b	16ms	55ms	28ms	44ms
B	0110b	65ms	210ms	86ms	131ms
C	1001b	260ms	900ms	260ms	390ms
C	1010b	1s	3.5s	1.8s	2.8s
D	1101b	4s	13s	5.4s	8.2s
D	1110b	17s	64s	38s	58s

48.4 AXI Bridge Module

48.4.1 Product Overview

48.4.1.1 Overview

The PCIe Core provides an AXI Bridging capability for directly adding a PCI Express link to an AXI system fabric. This significantly reduces the time to design PCI Express into an AXI-based SOC.

The AXI Bridge Module acts as a bridge between the standard AXI interfaces and the PCIe Core native interfaces. The bridge interconnects the AXI interfaces within an AMBA-embedded system with a remote PCIe link, as either a root complex port or as an endpoint port. The bridge supports up to two AXI interfaces, one for an AXI master, and one AXI bus shared for AXI Slave and DBI bus access to the native PCIe core.

The AXI master interface enables a remote PCIe device to read and write to an AXI slave connected to the AXI bridge. The AXI slave interface enables an AXI master to read and write through the AXI bridge to a remote PCIe device. The slave DBI (see DBI Access) enables an AXI master to read and write to registers inside the native PCIe core, It is shared with the AXI slave interface.

Throughout this chapter, the terms inbound and outbound are defined with respect to the AXI fabric. That is, inbound transactions are defined as the transactions presented by the native PCIe core's AXI master interface. Outbound transactions are defined as the transactions generated by an AXI master that targets a remote PCIe device.

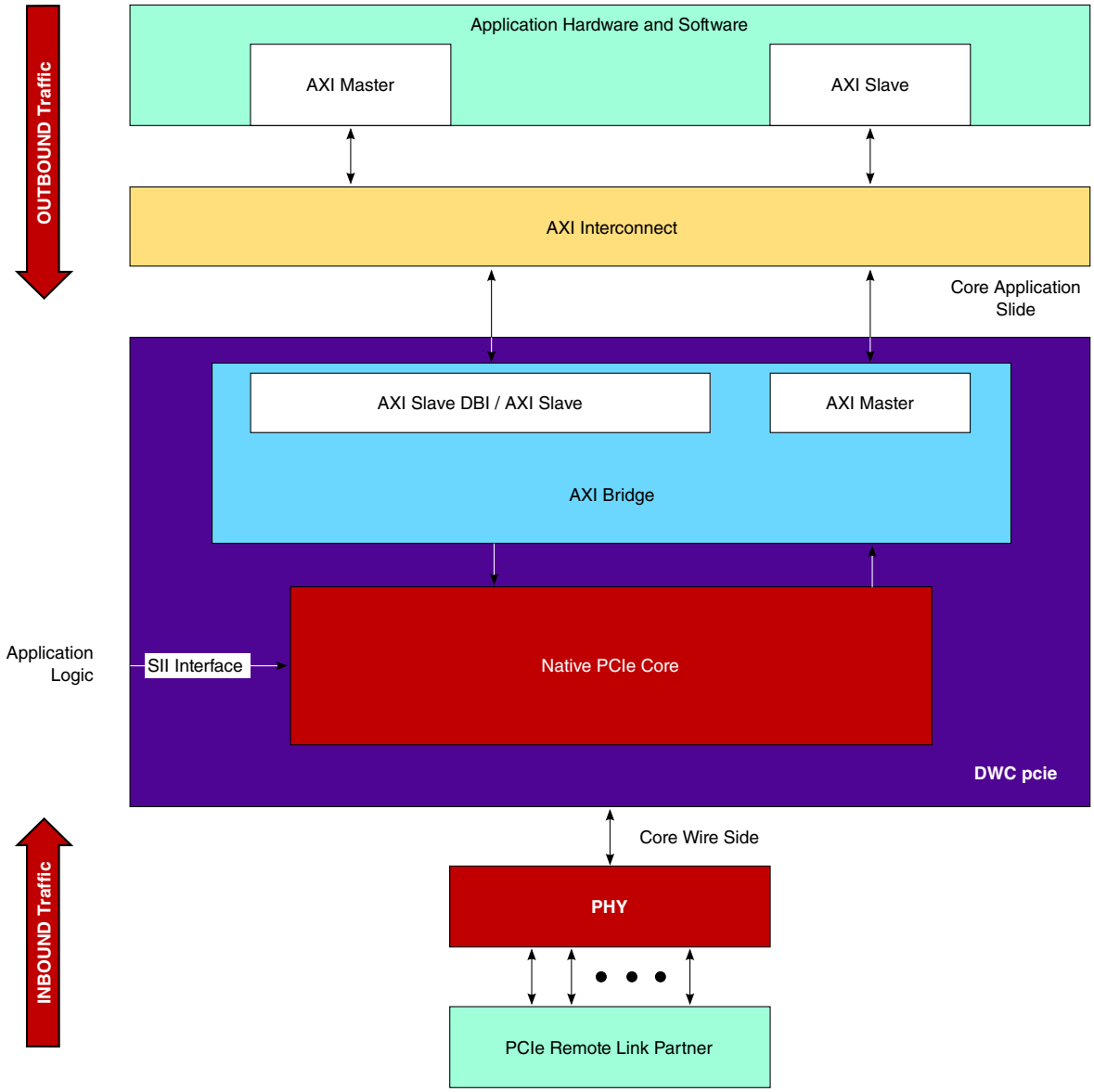


Figure 48-30. System-Level View of the DWC PCIe AXI Core

48.4.1.2 Interfaces

The figure below shows the DWC PCIe AXI core top-level interfaces.

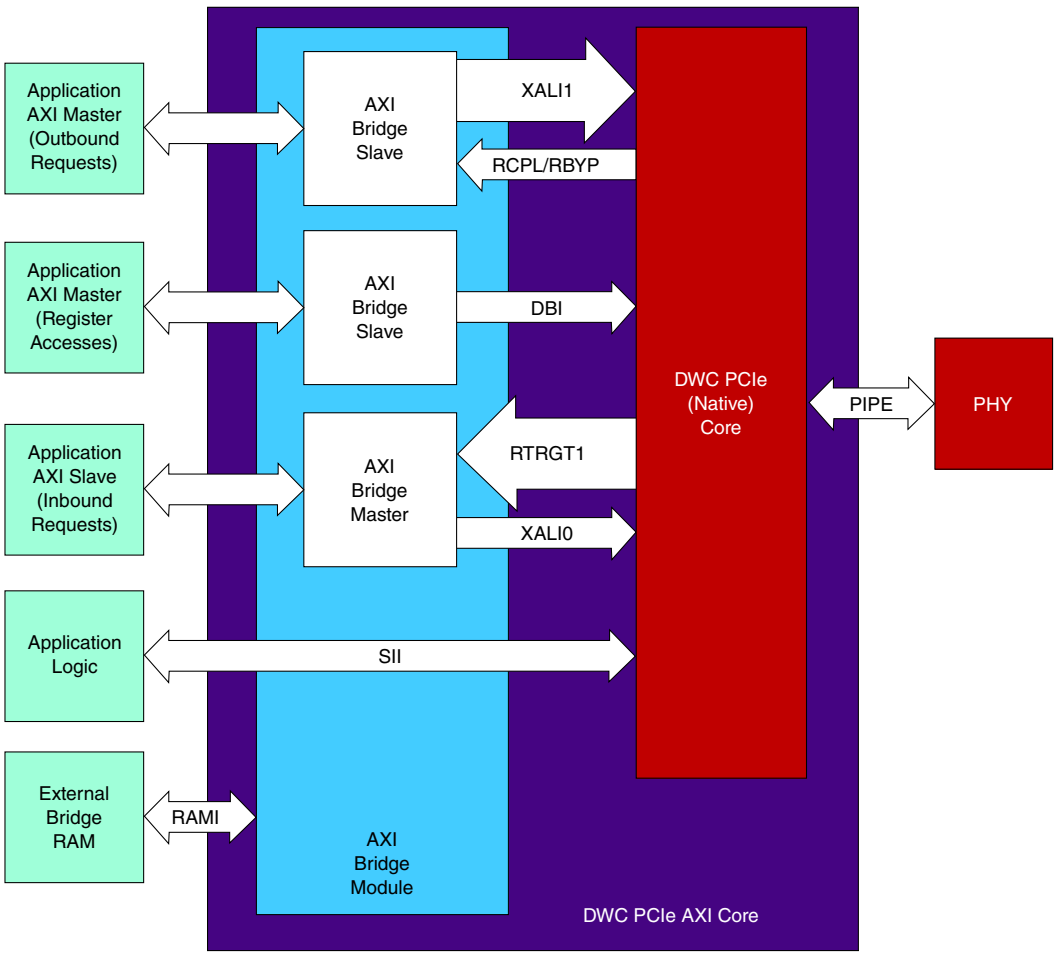


Figure 48-31. DWC PCIe AXI Core Top-Level Interfaces

For definitions of acronyms used for block and interface names, see Terms and Abbreviations.

48.4.1.3 Features List

The AXI Bridge Module supports the following features.

- AXI Master and Slave interfaces for inbound and outbound PCI Express requests.
- Multi-function support (up to 8 functions) [EP mode only].
- All types of PCI Express transactions supported through the AXI Bridge.
- A shared AXI Slave interface to access native core's CDM registers
- Programmable buffer sizes for AXI master and slave requests.

- Independent programmable user-defined clock rates for the PCI Express core, AXI master bus, AXI slave bus,.
- Programmable maximum number of outstanding inbound and outbound read requests for AXI.
- All burst-sizes supported for both AXI master and slave interfaces.
- Programmable and extended AXI burst lengths to support up to 4K read/write burst lengths over AXI master and slave interfaces.
- Little-endian operation.
- Independent maximum read request and transfer sizes between AXI and PCI Express (transfers can be split into multiple transfers).
- Response to AXI slave request combined gathering from split PCI Express completions that are received in-order.
- Response to AXI master request combined gathering from multiple AXI responses.
- PCIe legacy interrupt or MSI support.
- User-defined error mapping between PCI Express errors (UR, CA, CRS, poisoned, and ECRC error) and AXI slave response errors (SLVERR and DECERR).
- User-defined error mapping between PCI Express errors (UR, CA, CRS, poisoned, and ECRC error) and AXI master response error (DECERR_W and DECERR_R).
- Programmable byte parity check for the address and data buses throughout the bridge.
- Non-contiguous byte enables supported for inbound read/write and outbound write TLPs.
- Programmable MSI Interrupt controller to detect and terminate inbound MSI TLP's in the bridge for RC and DM (RC mode) products.

48.4.1.4 Limitations

The table below identifies the limitations encountered when using the AXI bridge with the DWC PCIe native core.

Table 48-25. AXI Bridge Limitations

AXI Bridge Limitation	Note
PCIe completions of a decomposed outbound AXI read request must be returned in-order.	Decomposition of outbound reads will not occur if your application master always generates read requests of size less than Max_Read_Request_Size. If decomposition occurs, and the remote device (or switch) is reordering completions, then corruption will occur at the Slave Response Composer in the AXI bridge. The Slave Response Composer only performs only in-order reassembly. In this case, you can set the AMBA Multiple Outbound Decomposed NP Sub-Requests Control Register to '0' to avoid corruption, at the expense of some performance.

Table continues on the next page...

Table 48-25. AXI Bridge Limitations (continued)

	See Outbound Ordering Limitation #1 for more details.
Posted and non-posted AXI writes targeting the bridge slave should use different AXI tags in order to allow the relevant B-channel responses to be received in order	See Outbound Ordering Limitation #2 for more details.
The AXI bridge only supports the CPL queue in ìStore and Forwardî mode, when a ìSegmented Queueî architecture is used. Otherwise, the CPL queue operates in Bypass mode, as it does for the ìSingleî and ìMultipleî queue architectures.	Set completions (CPL) to Bypass mode if you have not configured the core to use a segmented buffer queue architecture. See Queue to Port Mapping.
Vendor Messages must not be decomposed.	See AHB/AXI Message Address and Size Limitations for more details.
Big-endian operation is not supported	-
PCIe Advanced Error Reporting (AER) is not supported in the AXI bridge.	AER is only supported with respect to the native PCIe core. Errors detected by the bridge are not reported as part of AER.

48.4.2 Bridge Architecture

48.4.2.1 Bridge Architecture Overview

This module contains AXI master and slave protocol handlers, internal slave and master control for generic request and response interfaces, a packet composer, and a packet decomposer for response formation.

The slave and master protocol handlers support the AXI protocol conversion between an AXI transfer and a generic transfer within the bridge. The slave and master generic interface (GIF) supports the conversion of an AXI transfer to a PCIe transaction. The packet decomposer and composer support the segmentation and reassembly of a PCIe transaction.

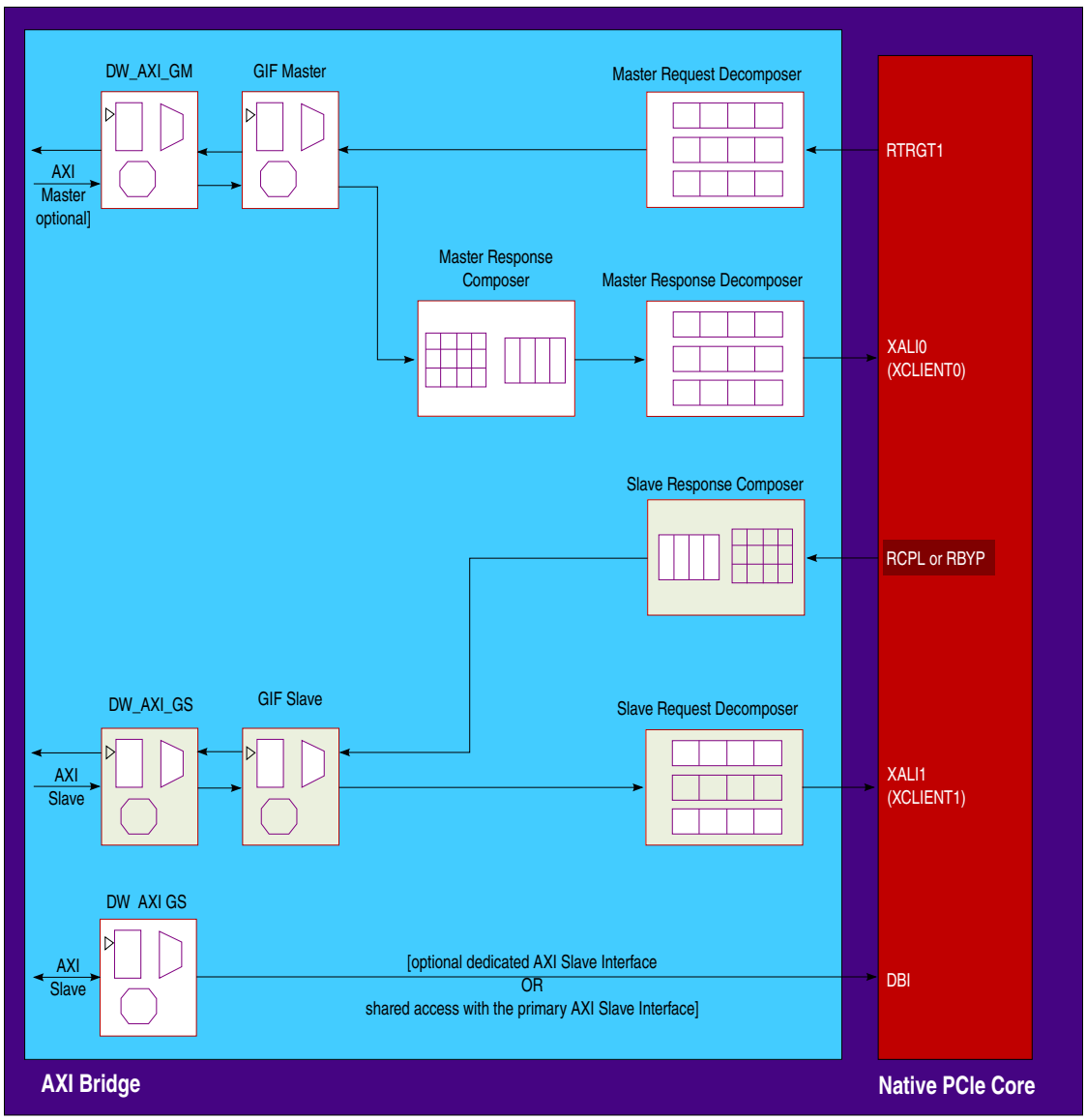


Figure 48-32. AXI Bridge Module, Block Diagram

The bridge module provides flexible buffering (Bridge Buffering) and tag management (Outbound Bridge Tag Management) to facilitate seamless bridging between the PCI Express native core and your application modules.

It has an optional master interface and a second optional slave (DBI) for accessing local registers in the native core CDM. Note also that DBI access is without the second slave (DBI). In this mode (shared DBI) the single slave is used to access both the DBI and the PCIe link.

48.4.2.1.1 Inbound Processing Module Chain

The following table lists the bridge modules involved in processing inbound requests.

Table 48-26. Bridge Modules Involved in Processing an Inbound (PCIe -> AXI) Requests

Module	Processing Step	Function
Master Request Decomposer	Request	Performs packet segmentation to satisfy the AXI fabric's "Max Burst Size" and "4KB Address Boundary" rules. These rules are discussed in Bridge Buffering.
Generic InterFace (GIF) Master	Request	The internal bridge circuitry is based on a generic protocol (GIF) and is not explicitly aware of whether it is interfacing to AXI or AHB.
Master Interface (DWC_axi_gm)	Request	This is the AXI protocol handler.
Master Response Composer	Response (Completion)	Performs packet reassembly caused by the bridge-initiated packet segmentation on the inbound request in the Master Request Decomposer.
Master Response Decomposer	Response (Completion)	Performs packet segmentation to satisfy PCI Express "Max Payload Size" and "Max Read Request Size" rules. These rules are discussed in Bridge Buffering.

48.4.2.1.2 Outbound Processing Module Chain

The following table lists the bridge modules involved in processing outbound requests.

Table 48-27. Bridge Modules Involved in Processing an Outbound (AXI -> PCIe) Requests

Module	Processing Step	Function
Slave Interface (DWC_axi_gs)	Request	This is the AXI protocol handler
Generic InterFace (GIF) Slave	Request	The internal bridge circuitry is based on a generic protocol (GIF) and is not explicitly aware of whether it is interfacing to AXI or AHB.
Slave Request Decomposer	Request	Performs packet segmentation to satisfy PCI Express ìMax Payload Sizeî and ìMax Read Request Sizeî rules. These rules are discussed in Bridge Buffering.
Slave Response Composer	Response (Completion)	Performs packet reassembly caused by the following sources of segmentation: •Bridge initiated packet segmentation on the outbound request in the Slave Request Decomposer. •Packet segmentation caused by the remote completer.

The application master request (at the AXI Slave Interface) will not - AXI protocol demands it - request more than 128 bytes and will not issue a request that involves crossing a 4K page boundary. Therefore the AXI response (completion) - composed from one or more completions that arrive in off the PCIe wire -is AXI compliant and does not need to be processed in a decomposer to enforce AXI page boundary and maximum packet size rules. Therefore there is no Slave Response Decomposer in the bridge module. In contrast, there is a Master Response Decomposer.

NOTE

CC_SLV_MTU

Also known as CC_SLV_RD_REQ_SIZE

This value is used to set memory sizes in the bridge (Slave Request Decomposer Data FIFO and Slave Response Composer).

Calculated as CC_SLV_BURST_LEN * (SLAVE_BUS_DATA_WIDTH / 8) = 16 * 64 / 8 = 128

CC_SLV_BURST_LEN is the maximum burst in beats that the application will ever issue to the bridge slave interface.

48.4.2.2 Decomposition

The bridge manages the different transfer sizes between PCIe and AXI.

The maximum payload is configured independently between the AXI and PCIe systems. Each system's transfer size will sometimes need to be converted causing a split of a request and composition of a response.

Automatic segmentation and reassembly of outbound (application to PCIe wire) packets is performed to satisfy PCI Express *Max_Read_Request_Size* and *Max_Payload_Size* rules. When a decomposition rule is triggered, the outbound TLP is broken up into two or more smaller TLPs. If the TLP type is Non Posted then additional PCIe TAGs are used from the TAG pool. Additional entries in the outbound header buffer are also used.

In a similar manner, automatic segmentation and reassembly of inbound packets is performed. The same AXI ID is used for all decomposed Non-Posted (but not for Posted; see Inbound Bridge Tag Management) packets.

The host software can program the PCIe device to support certain maximum write transfer sizes and maximum read request sizes. The native PCIe core's configuration module contains the device's *Max_Payload_Size* and *Max_Read_Request* size information, which are defined by the application software. The AXI bridge supports mismatches that occur when the AXI maximum transfer length is different than the *Max_Payload_Size* and *Max_Read_Request* size. For example, an inbound read transfer has an associated response buffer that is dependent on the remote PCIe device's *Max_Read_Request* size. An inbound write transfer has a master write buffer that is dependent on the remote PCIe device's *Max_Payload_Size*.

More detailed information on decomposition rules and effects is available at the end of the AXI bridge section at AXI Decomposition Rules.

48.4.2.3 Bridge Buffering

The bridge has additional buffering in addition to buffering already present in the native core.

The following table identifies all of the RAMs that are used in the AXI bridge for buffering purposes.

Table 48-28. Lists of Buffering Related RAMS

RAM Name	Alternative Name
Slave Request Decomposer Header and Data Queues : buffering of Outbound AXI Request	XADMX1 Decomposer Header and Data RAMs
Slave Response Composer : buffering of Inbound Responses to Outbound AXI Request	RADMX Composition RAM
Slave Response Asynchronous Clock Crossing FIFO : synchronization of Inbound Responses to Outbound AXI Request	RADMX Asynchronous RAM
Master Request Decomposer Header and Data Queues : buffering of Inbound PCIe Request	RADMX Decomposer Header and Data RAMs
Master Response Composer : buffering of Outbound Responses to Inbound PCIe Request	GM Composition RAM
Master Response Decomposer Header and Data Queues : buffering of Outbound Responses to Inbound PCIe Request	XADMX0 Decomposer Header and Data RAMs

More detailed information on buffering is available at the end of the AXI bridge section at Outbound Bridge Tag Management .

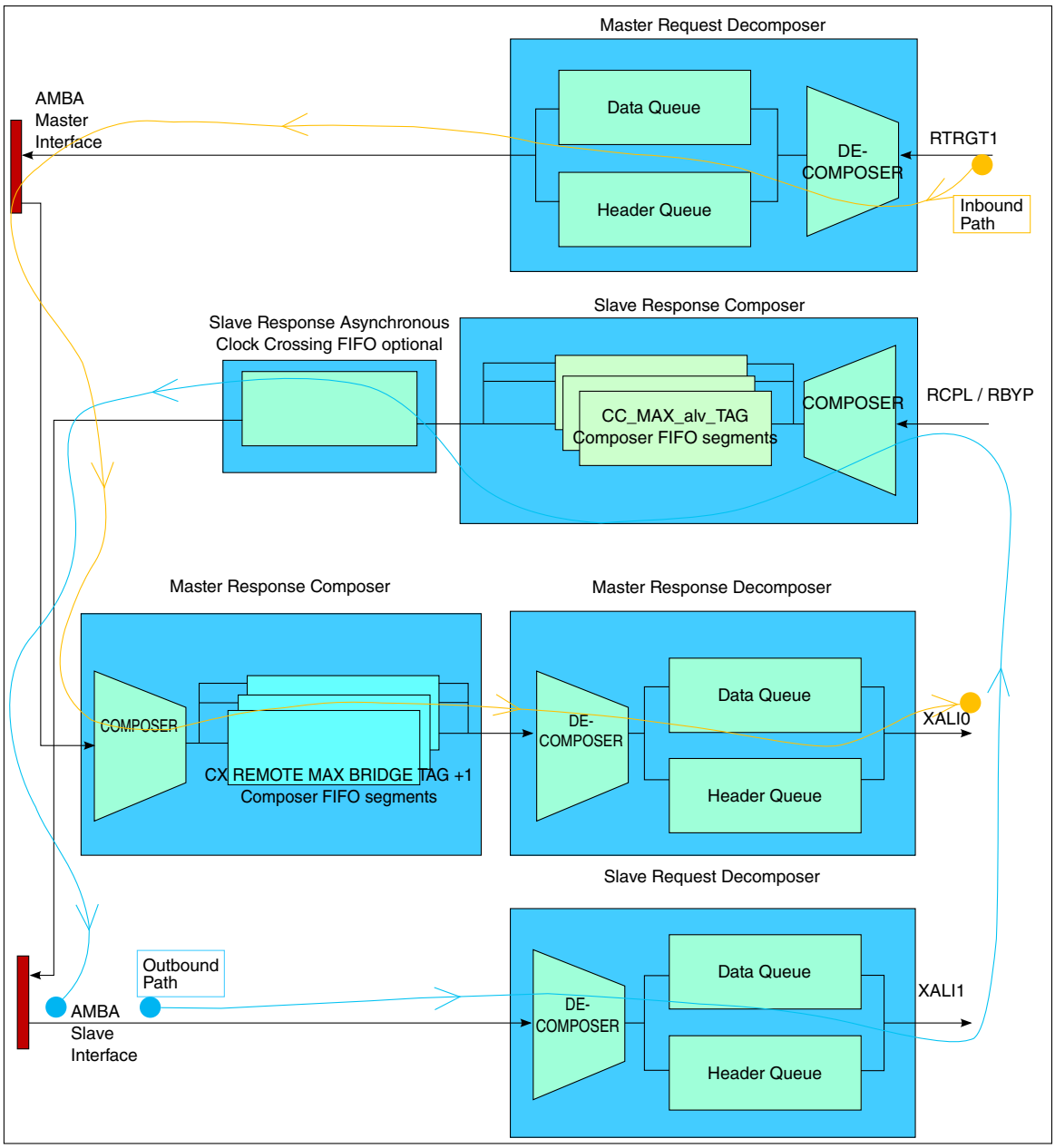


Figure 48-33. Overview of Bridge Buffering

48.4.2.4 Outbound Bridge Tag Management

The bridge transfers IDs/TAGs between PCIe and AXI.

Transactions on PCIe and AXI bus have separate IDs/TAGs and as the bridge transfers the data, it maps and keeps track of the IDs/TAGs on each side.

The diagram found here gives an overview of the outbound tag management system for Non Posted requests.

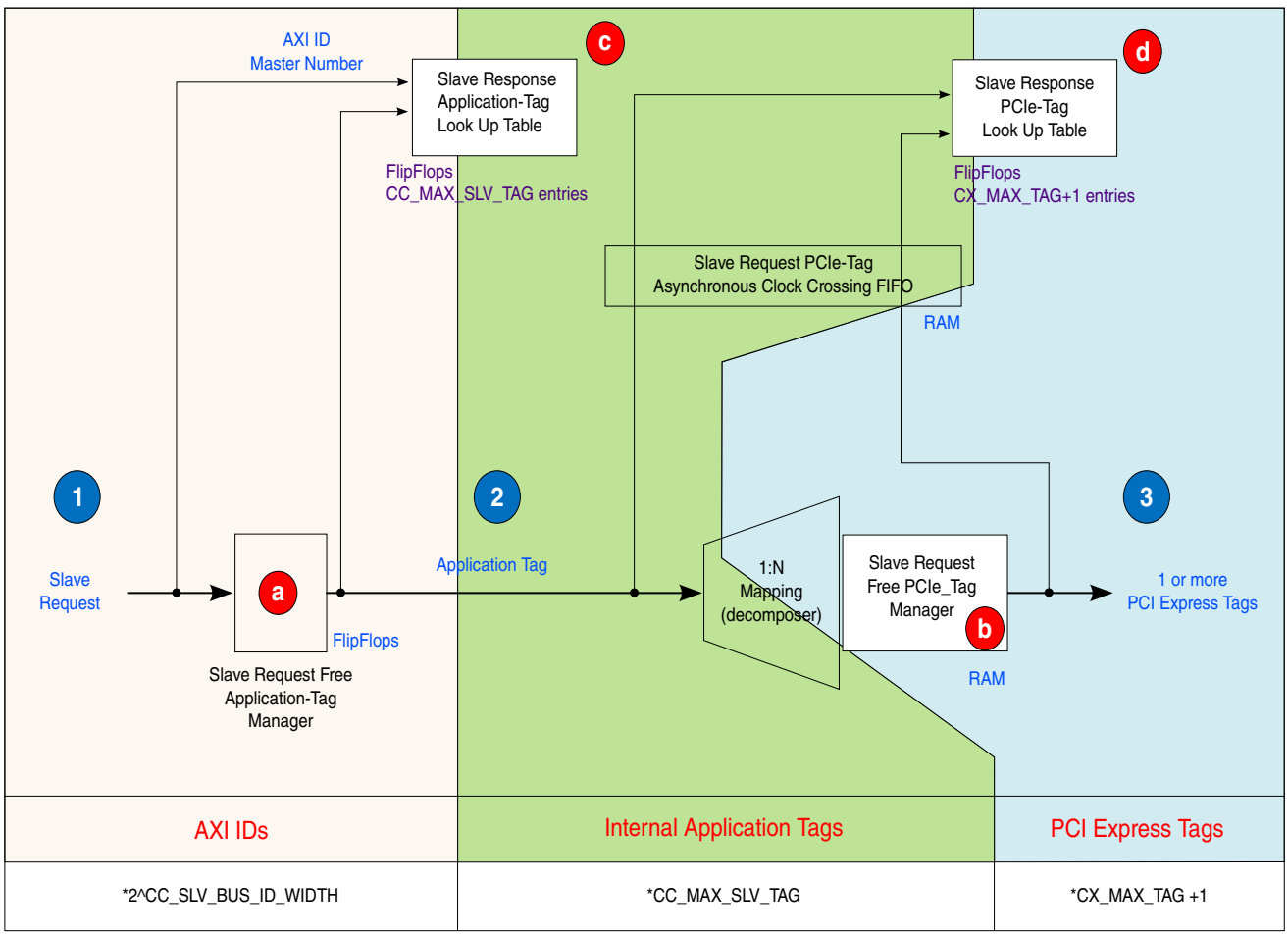


Figure 48-34. Outbound Tag Management Architecture (Non Posted requests only) - request path is shown but completion path is omitted.

When an AXI outbound Non Posted request at the slave interface is converted to a PCIe request TLP, it is necessary to:

- Select a PCIe tag from the 'free' pool of currently available PCIe tags. This is performed using a Tag Manager (see Module 'b' in figure above). When the pool is exhausted, no more PCIe request TLPs can be transmitted until the pool is replenished by tags from inbound PCIe completion TLPs. See steps 2 and 3 in figure above.
- Store the mapping from AXI request to PCIe TAG, so that when the corresponding PCIe completion TLP arrives back in the future, it can be correctly associated with

the originating request. This is achieved using a Tag Look Up Table (LUT) - see Module 'd' in figure above.

The process of mapping AXI requests IDs to PCIe tags is a two-step process - Application Tag mapping and PCIe Tag mapping (as described above).

Application Tag mapping -which is internal to the bridge -is an intermediate step which has to occur first. See steps 1 and 2 in figure above.

In the AXI protocol, it is possible to have several AXI requests with the same AXI ID. Therefore it is necessary for the bridge to differentiate between individual AXI requests to enable it to track them and their completions throughout the system.

This is necessary to aid in reassembling or recomposing (but not necessarily re-ordering, see Outbound Ordering Limitation #1 for more details) the multiple inbound completions that are returning for an original single AXI outbound request.

Multiple completions can occur due to decomposition (see Decomposition) of the original outbound request or due to the remote link partner choosing to complete a request using multiple completions.

Application tag mapping occurs via the following process:

- The bridge maps the AXI ID to an internal application tag (see Module 'a' in figure above). This application tag identifies the reserved space in the Slave Response Composer for the expected returning completions.
- The bridge store the mapping from the AXI ID to the internal application tag, so that when the corresponding PCIe completion TLP arrives back in the future, it can be correctly associated with the originating request. This is achieved using a Tag Look Up Table (LUT) - see Module 'c' in figure above.
- If decomposition occurs, each PCIe TLP is assigned a unique PCIe tag from the pool of CX_MAX_TAG (31) tags (see Module 'b' in figure above). Each of these PCIe tags is associated with the same internal application tag and this information is stored is a LUT (see Module 'c' in figure above).

If at any time, all PCIe TAGs or internal application tags are consumed, then the AXI Bridge Slave interface will not accept any new transactions and will apply back-pressure to the AXI fabric.

48.4.2.5 Inbound Bridge Tag Management

The bridge transfers IDs/TAGs between PCIe and AXI.

Transactions on PCIe and AXI bus have separate IDs/TAGs and as the bridge transfers the data, it maps and keeps track of the IDs/TAGs on each side. When an inbound PCIe request TLP is converted to an AXI request at the master interface, it is necessary to:

- Select an AXI ID from the 'free' pool of currently available AXI IDs.
  - For Non Posted requests, this is done using an ID Manager - see Module 'b' in the figure below.
  - Posted (P) requests always use ID '0'.
- When the pool is exhausted, no more requests can be launched onto the AXI fabric until the pool is replenished by IDs. An ID is released to the pool when the completion has arrived back from the application AXI master. In the case where the inbound PCIe TLP request was decomposed, then the ID is released when all the individual completions have arrived back into the master response composer.
- Store the mapping from PCIe TAG to AXI request ID to, so that when the corresponding AXI response arrives back in the future, it can be correctly associated with the originating request. This is achieved using an ID Look Up Table (LUT) - see Module 'a' in the figure below.

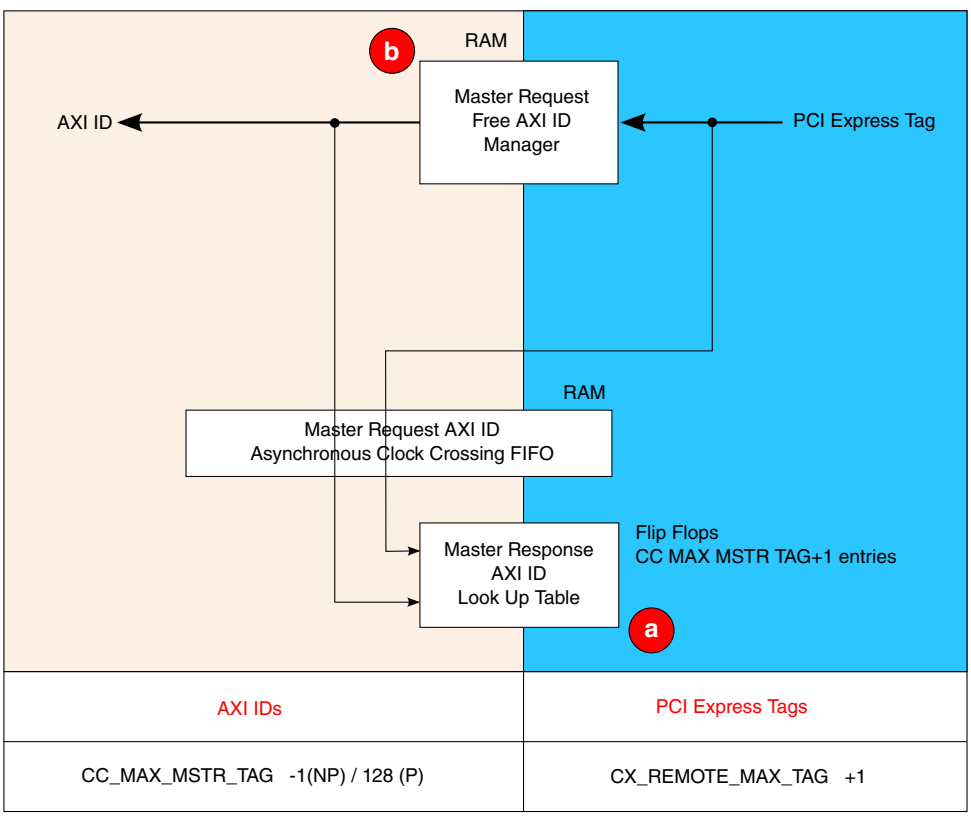


Figure 48-35. Inbound Tag Management Architecture - request path is shown but completion path is omitted.

Decomposition (Decomposition) does not affect AXI ID usage. If decomposition is occurring, then the same ID is used for each AXI request that is generated by decomposition. Each response is identified by the AXI bridge using its response ID.

Since the PCIe Posted transfer expects no response (completion) for a Posted request, the AXI bridge drops the write responses.

The AXI bridge master interface issues all Posted requests with an ID of '0'. The AXI bridge master interface issues Non Posted requests with an ID from the range 1 to CC_MAX_MSTR_TAG -1 (4-1=3).

It is possible to have CC_MAX_MSTR_TAG-1 (4-1=3). outstanding Non Posted requests and 128 Posted requests at the same time since Posted requests are always writes and use a different AXI ID signal (mstr_awid). The following memory devices are connected with inbound tag management in the bridge.

48.4.2.6 Inbound Order Enforcement for AXI Bridge

The DWC PCIe AXI bridge has the inbound buffers shown in the figure found here.

- a single FIFO queue structure (Master Request Decomposer Data and Header Queues) for inbound read/write requests
- a segmented buffer queue structure (Slave Response Composer) for inbound completions in response to outbound read requests.

There is no reordering after an inbound request has left the native core's receive queue. The request FIFO is blocking and will not allow read or write requests pass each other. This default design architecture is designed to serve inbound traffic as a FIFO (first-come, first -served) i.e. in-order service.

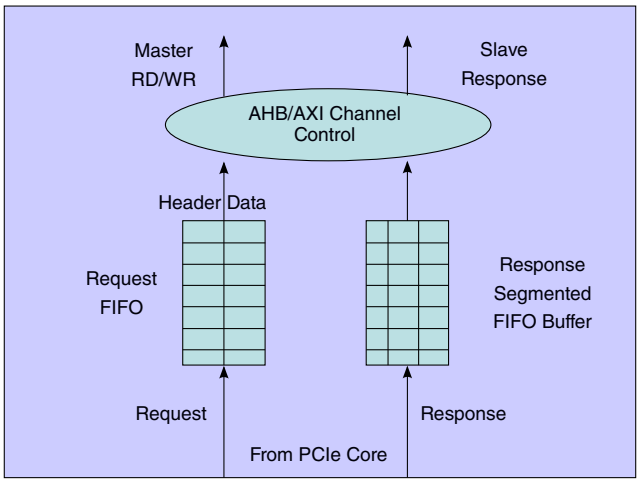


Figure 48-36. DWC PCIe AXI Bridge Inbound Traffic Queue Architectures

The native core (before the bridge) has flexible buffering (see Queue to Port Mapping and PCIe order rule enforcement is performed within the native core. See PCIe Core Inbound Order Enforcement.

48.4.2.6.1 Ordering Enforcement Hardware Lock Feature

Once a completion has been taken out of the receive queues of the DWC PCIe core, it can pass a posted request.

To enforce the PCIe Ordering rule ìcompletions should not pass posted if relaxed ordering is not setî for applications that require it, the DWC PCIe AXI bridge employs a hardware lock mechanism feature.

This feature will prevent completions from passing posted requests by requiring that posted transactions all complete through the bridge to an AXI interface before a completion can be taken out of the receive queue. This ensures that previous posted transactions are never passed by a completion. This feature is turned on when the DWC PCIe core is configured to have a receive queue in the segmented buffer mode and completion is in store-forward mode.

The feature employs packet halt signals generated by the bridge and used by the core when deciding how to unload the segmented buffer queue of the core. There is an individual halt signal for posted, non posted and completion transactions. Completion transactions are halted when there are any posted transactions in the bridge FIFO.

The limitation of the above feature is that it gives the posted transactions priority such that it can always pass through the bridge when it is available, while the completions are blocked.

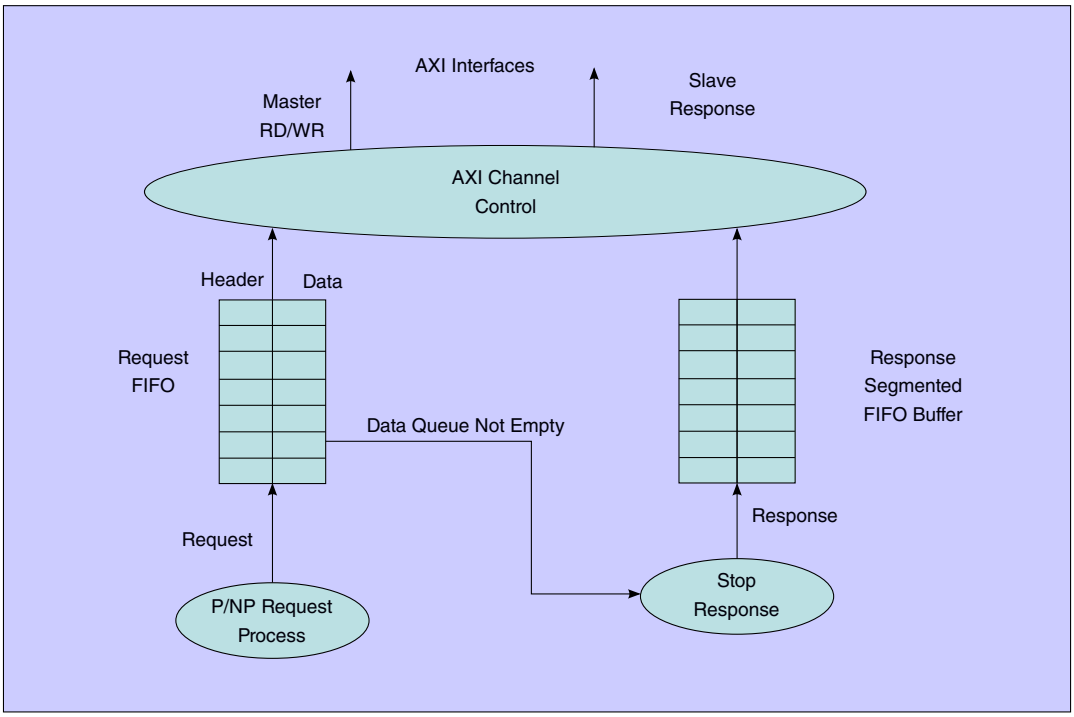


Figure 48-37. DWC PCIe AXI Bridge Inbound Traffic Order Enforcement

48.4.2.6.2 Re-Ordering Effects of AXI Fabric

The ordering rules, in typical AXI bridge configurations, apply only to the PCIe native core inbound queue and not after the inbound or outbound traffic has reached the AXI bridge. This means, for example, if a posted request followed by a completion reaches the AXI bridge, then it is possible that the completion will reach the AXI slave response channel before the posted reaches the AXI master request channel. This depends on the readiness of AXI master request channel (if wait states are asserted by the AXI bus).

The AXI bridge has independent AXI channels for outbound and inbound transfers. Therefore, a read response can be presented on the read outbound AXI channel while a write request is being presented onto the write inbound AXI channel. Since the response of an outbound read and inbound write request can be presented onto the application logic of AXI simultaneously, then the PCIe traffic order rules do not get completely enforced by most AXI bridge applications.

The AXI bridge master will generate a unique ID for each Non-Posted request (If decomposition is occurring, then the same ID is used for each AXI Non Posted request. For more details see Decomposition and Inbound Bridge Tag Management). Each response is identified by the AXI bridge using its response ID.

The AXI bridge master interface issues a Posted request (write) with a unique ID that is independent from master reads. Since the PCIe Posted transfer expects no response, the AXI bridge drops the write responses. If decomposition is occurring, then a unique ID is used for each AXI write request. For more details see Inbound Bridge Tag Management.

48.4.2.6.2.1 Inbound Ordering Limitation

Sometimes a Non-Posted transaction can pass a Posted transaction on the AXI fabric and this is a violation of the ordering rules. The AXI master interface does not wait for a write response - when it issues write transactions on the AW channel - before issuing a read transaction issued on the AR channel. As a result, a read transaction issued on the AR channel (after a write transaction on the AW channel) could complete first. This could happen if the AXI fabric delays completion of a write request (for example, due to latency in a write buffer) and a read request from the same address follows the write.

NOTE

For information on AXI ID generation see Inbound Bridge Tag Management.

48.4.2.6.3 Additional Information (ordering)

- Order enforcement through the native core is discussed in App Note: Order Enforcement Using the PCIe Core.
- Outbound Order Enforcement for AXI Bridge

The DWC PCIe AXI bridge has the following outbound buffers (see the figure below):

- a single FIFO queue structure (Slave Request Decomposer Data and Header Queues) for outbound read/write requests
- a segmented buffer queue structure (Master Response Composer) for outbound completions in response to inbound read requests.

These two queues are independently structured. Therefore, there is no order maintained (between completions and requests) once outbound requests or outbound responses have entered the bridge.

The FIFO is blocking and will not allow read or write requests pass each other. This default design architecture is designed to serve outbound traffic as a FIFO (first-come, first -served) i.e. in-order service. Outbound requests are served onto the PCIe wire in the order that they are received from an AXI master.

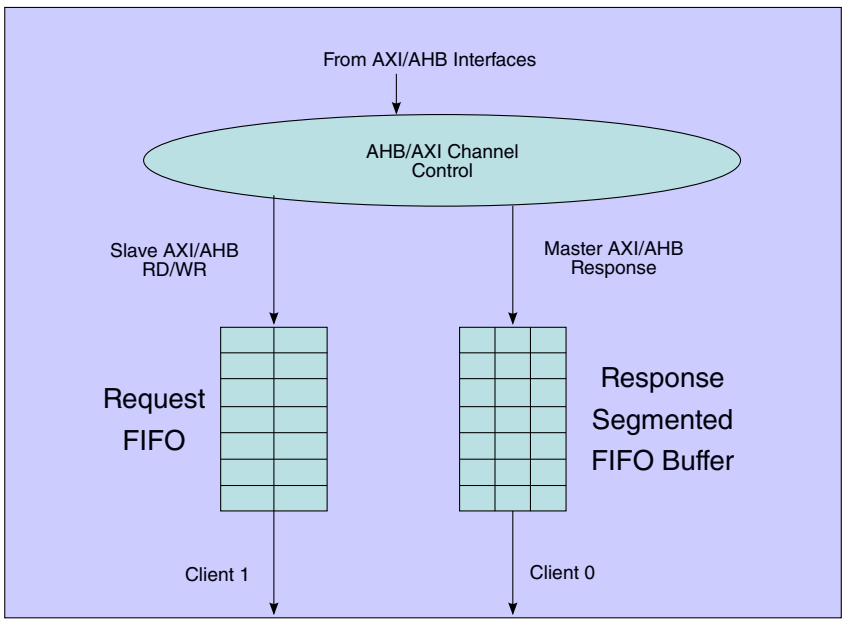


Figure 48-38. AXI Bridge Outbound Traffic Queuing Architecture

The native core (after the bridge) has no transmit buffering and requests and completions are transmitted onto the wire directly. Completions use a different interface (XALI0) to requests (XALI1) and may or may not pass the outbound requests depending on client arbitration.

48.4.2.6.4 PCIe Completion Reordering

If the returning completions (for all Non Posted outbound requests with the same AXI ID) from the remote device are returned out-of-order, the bridge Slave Response Composer will re-order them according to AXI IDs.

Only completions corresponding to outbound requests with the same AXI ID are re-ordered, while all others ones will be returned following the chronological order in with which they are received from the remote device.

48.4.2.6.5 Outbound Ordering Limitation #1

An outbound AXI request at the slave interface may be decomposed into multiple PCIe TLP requests.

See Decomposition for more information. There is a PCI Express protocol ordering rule that states that the completions associated with the different TLPs may pass each other. The current bridge Slave Response Composer design cannot handle this scenario and will not re-order the completions, resulting in data corruption. To avoid data corruption, one of the following workarounds is required.

- Prevent outbound decomposition by ensuring that your application master does not generate bursts of size greater than Max_Read_Request_Size. Otherwise, you must program or design your PCI Express system with a larger value of Max_Read_Request_Size. See Decomposition Side-Effects.
- Set AMBA Multiple Outbound Decomposed NP Sub-Requests Control Register to '0' which will disable the possibility of having multiple outstanding non-posted requests that were derived from decomposition of an AMBA request. This should only be done when decomposition is guaranteed to occur and completions are coming back out of order from the wire, as it restricts the PCIe transmit link performance by only having one outstanding non-posted request TLP (from a request that is being decomposed) on the PCIe link at any one time.

It is also a PCI Express ordering rule that a remote device which completes a single request using multiple completions (CplD) must return them in-order. The bridge can accept an interleaved stream of multiple completions (in response to multiple requests it originally transmitted), provided that all the completions for any one original request are in-order.

48.4.2.6.6 Outbound Ordering Limitation #2

Posted and non-posted AXI *writes* targeting the bridge slave must use different AXI tags in order to allow the relevant B-channel responses to be received in-order.

If this restriction is not observed, it can happen that a B-channel response to a posted (P) write (MWr/Msg) arrives before the B-channel response associated to a non posted (NP) write (for example, IOWr), even if the NP request send first.

The B-channel response to a P write is issued by the AXI bridge slave immediately after the request is accepted by the bridge slave. The B-channel response to an NP write is issued by the AXI bridge slave when the completion (CplD) arrives back from the remote link partner.

48.4.2.6.7 Additional Information (AXI bridge bandwidth)

Order enforcement through the native core is discussed in App Note: Order Enforcement Using the PCIe Core.

48.4.3 PCIe AXI Core Operations

48.4.3.1 AXI Sideband (Misc. Bus) Signals

An AXI sub-system can only convey data, address and read/write information. It has no way to propagate PCIe concepts such as TLP type, Posted/Non Posted, Function Number, TLP attributes and so on. Therefore, some method is required to map these PCIe concepts between AXI and PCIe. This is done through the Software option.

Use the Internal Address Translation (iATU) to map outbound AXI transactions from different AXI address regions to particular PCIe address regions and TLP types. For example, map all AXI transactions in the region 0x0010FD00 - 0x00FFFFFF to CFG TLPs in the PCIe address region 0xD010FD00 - 0xD0FFFFFF.

48.4.3.2 Supported AXI Transfer Type

The AXI Bridge Module is compliant with the AMBA 3.0 AXI specification.

48.4.3.3 Supported AXI Burst Operations

- For outbound transfers (accessing bridge SLAVE):
  - The AXI bridge slave supports the incremental burst type (INCR) which is used in conjunction with ARLEN and AWLEN to define any length of burst.
  - The AXI bridge slave does not support the WRAP and FIXED burst types. If your application issues these burst types to the AXI bridge slave, a SLVERR or DECERR response is not returned by the bridge slave. In this scenario, the PCIe core exhibits undefined behavior.
- For inbound transfers (using bridge MASTER):
  - The AXI bridge master used the incremental burst type (INCR) which is used in conjunction with ARLEN and WLEN to define any length of burst.
    - The parameter CC_MSTR_BURST_LEN controls the maximum length burst that will be generated by the master interface.
  - The AXI bridge master does not use the WRAP and FIXED burst types.

48.4.3.4 I/O and CFG Transaction Handling over AXI Bridge

I/O and CFG-type inbound and outbound transfers are fully supported by the AXI bridge. I/O and CFG transactions always have the following characteristics:

Table 48-29. IO and CFG Characteristics

A payload of length equal to one DWORD (four bytes, 32 bits).
First byte enable (FBE) can be any value including '0000'.

Table continues on the next page...

Table 48-29. IO and CFG Characteristics (continued)

A payload of length equal to one DWORD (four bytes, 32 bits).
Last byte enable (LBE) is always '0000'.

48.4.3.4.1 Outbound I/O and CFG Transaction Handling

The following conditions must be satisfied for all outbound CFG and I/O requests presented by the application to the bridge slave interface:

- The outbound IO/CFG transactions should be DWORD aligned. Therefore, slv_a*addr[1:0]=00b and slv_a*size=2.
- The outbound AXI -> PCIe transfer must complete in a single AXI transfer without the need for a bus burst or a series of bus transfers.
- The maximum burst length for outbound IO and CFG transfers is one, corresponding to one DWORD.

There is one methods that can be used to signal to the PCIe core that the application wants to send an IO or CFG transfer.

48.4.3.4.1.1 Method I: Address Translation Method of Sending an Outbound IO or CFG Transfer

The optional internal address translation unit (iATU) in the native core can change the TYPE of an outbound request from MEM to IO or CFG by matching the address of that request to a configured address range as set by the application. See Internal Address Translation (iATU).

NOTE

You should use the ìInternal Address Translation (iATU)î instead of the Slave Request Sideband Bus (slv_a*misc) - to transmit CFG or IO requests.

It is also possible to do the same (for READs only) using a customer defined external address translation unit.

48.4.3.4.2 Inbound I/O and CFG Transaction Handling

The inbound PCIe -> AXI transfer completes in a single transfer without the need for a burst or a series of transfers except during:

- NCBE transfer. e.g. FBE = 1011 | 0101 | 1001 | 1101.
- Unaligned[1] transfer. e.g. FBE = 1110 | 1100 | 1000
- Narrow transfer. e.g. FBE = 1110 | 1100 | 1000 | 0111 | 0011 | 0001

1.For an unaligned (i.e. non-DWORD aligned) inbound PCIe -> AXI transfer (FBE = 1110 | 1100 | 1000), the AXI transfer still completes in one bus beat since LBE = 0000.

48.4.4 Additional AXI Reference Material

48.4.4.1 AXI Decomposition Rules

The host software can program the PCIe device to support certain maximum write transfer sizes and maximum read request sizes.

The native PCIe core's configuration module contains the device's *Max_Payload_Size* and *Max_Read_Request* size information, which are defined by the application software. The AXI bridge supports mismatches that occur when the AXI maximum transfer length is different than the *Max_Payload_Size* and *Max_Read_Request* size. For example, an inbound read transfer has an associated response buffer that is dependent on the remote PCIe device's *Max_Read_Request* size. An inbound write transfer has a master write buffer that is dependent on the remote PCIe device's *Max_Payload_Size*.

48.4.4.1.1 Outbound Decomposition

Automatic segmentation and reassembly of outbound (application to PCIe wire) packets is performed to satisfy PCI Express *Max_Read_Request_Size* and *Max_Payload_Size* rules. When a decomposition rule is triggered, the outbound TLP is broken up into two or more smaller TLPs. If the TLP type is Non Posted then additional PCIe TAGs are used from the TAG pool. Additional entries in the outbound header buffer are also used.

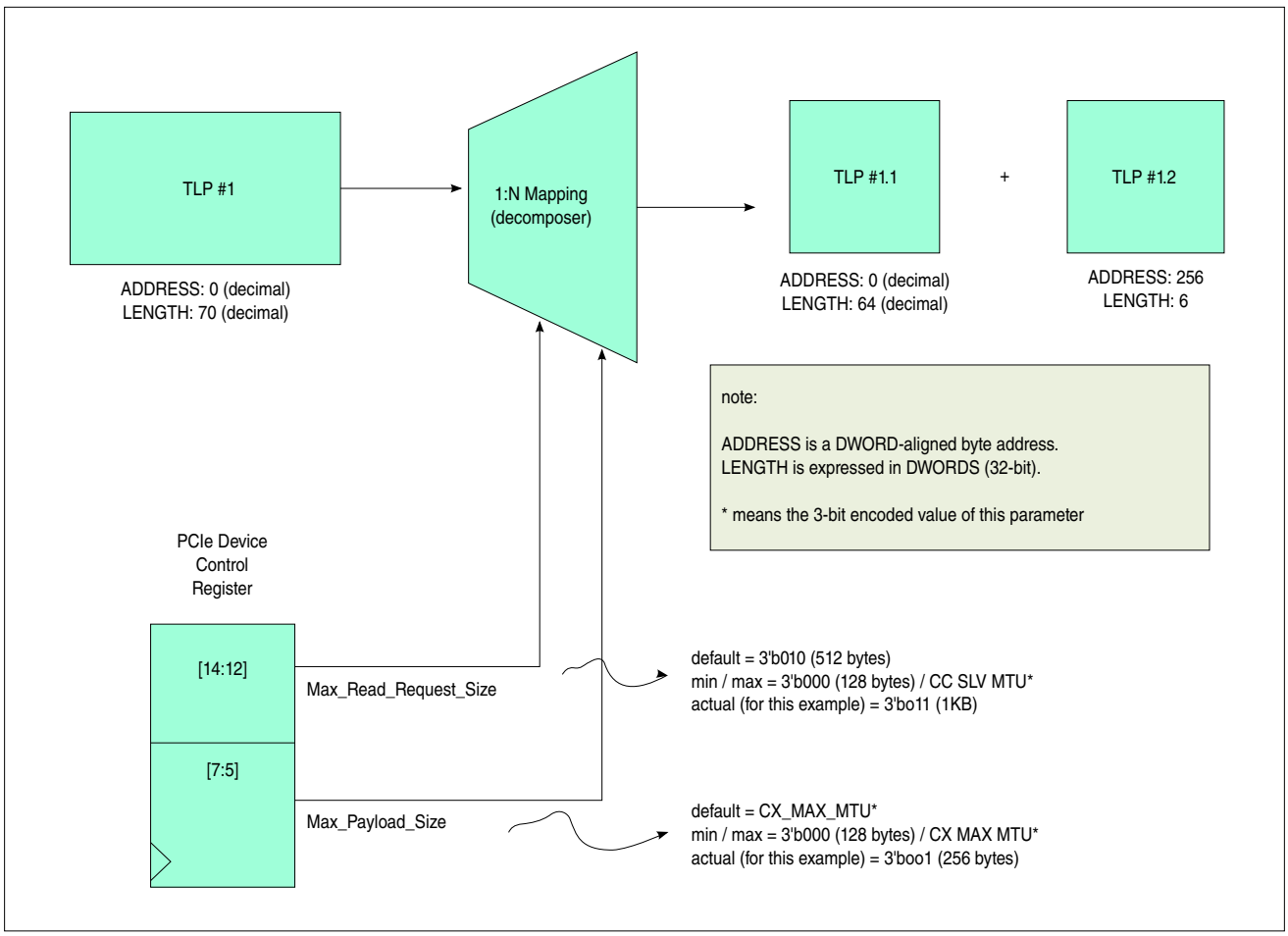


Figure 48-39. Outbound Decomposition Example of MemWr with Max Payload Size set to 256 bytes (64 DWORDs)

The Slave Request Decomposer module uses the following rules to determine when decomposing takes place.

Table 48-30. Outbound Decomposition Rules

TLP Request Type	Decompose[1] When
Read	TLP length >= PCIe Max_Read_Request_Size
Write	TLP length >= PCIe Max_Payload_Size

1. To avoid possible decomposition when the address is not DWORD aligned, you must ensure that TLP length is not only > Max_Read_Request_Size but also >= Max_Read_Request_Size.

The AXI page boundary is 4K, and so the application master request (at the AXI Slave Interface) will not issue a request that involves crossing a 4K PCIe page boundary - AXI protocol demands it.

NOTE

CX_MAX_MTU - 128

The largest packet payload (Maximum Transfer Unit) that the device will support. Specified in bytes and not DWORDS.

This is distinct from the maximum operating payload (Max_Payload_Size) which may be set by software.

48.4.4.1.1.1 Decomposition Side-Effects

Decomposition degrades the PCIe link performance as because it increases the amount of TLP header overhead and uses up extra PCIe TAGS.

Decomposition uses up extra header FIFO locations which will reduce the bridges bus-offloading ability in some cases.

48.4.4.1.1.2 Reducing Outbound Decomposition

To reduce decomposition on the outbound path, observe the following guidelines.

For Write Requests:

- Ensure that your application master does not generate bursts of size greater than or equal to Max Payload Size. In this context 'burst size' refers to the total number of bytes requested, or
- Program your PCI Express system with a larger value of Max_Payload_Size without exceeding CX_MAX_MTU (128).

For Read Requests:

- Ensure that your application master does not generate bursts of size greater than or equal to Max_Read_Request Size or,
- Program your PCI Express system with a larger value of Max_Read_Request without exceeding CC_SLV_MTU (128).

48.4.4.1.2 Inbound Decomposition

In a similar manner, automatic segmentation and reassembly of inbound packets is performed. The same AXI ID is used for all decomposed packets. The figure below illustrates the rules that are used in the Master Request Decomposer module to determine when decomposing takes place.

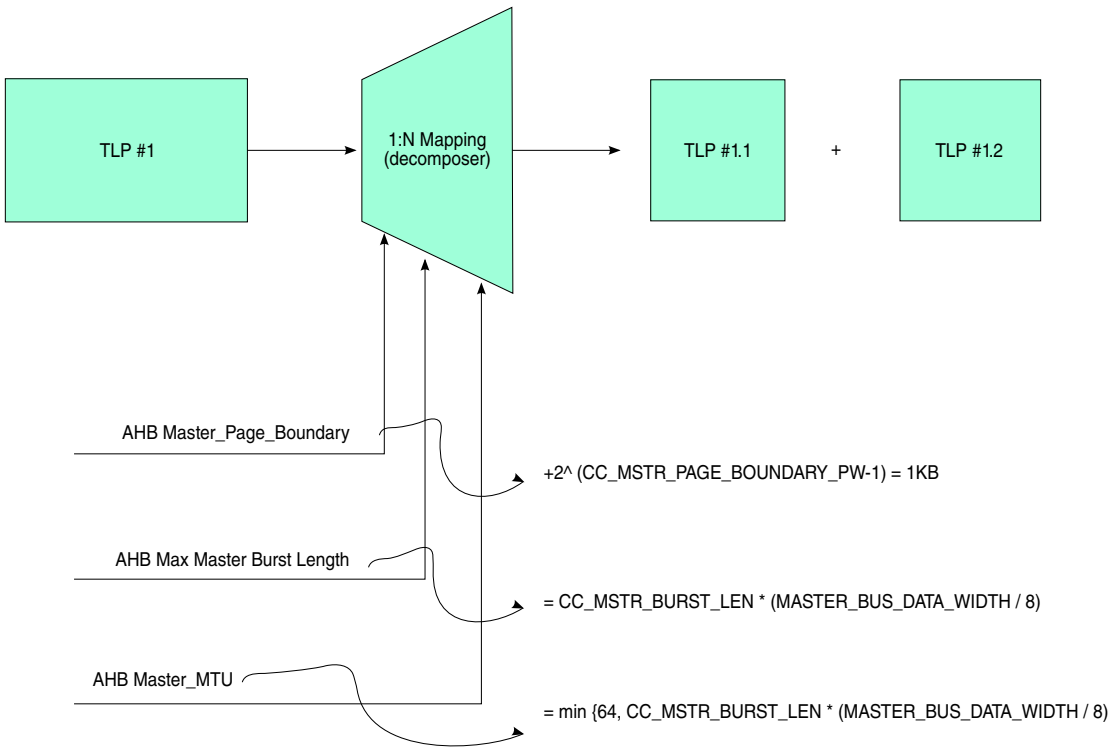


Figure 48-40. Inbound Decomposition

CC_MSTR_BURST_LEN (16) is the maximum burst in bus beats that the bridge will ever issue at the bridge master interface. It is set by the user.

The following rules are used in the Master Request Decomposer module to determine when decomposing takes place.

Table 48-31. Inbound Decomposition Rules

TLP Request Type	Decompose When
Read	TLP length (bytes) > AXI Master_Max_Read_Request_Size (see Figure 48-40)
Write	TLP length (bytes) > AXI Master_MTU (see Figure 48-40)
Read or Write	TLP address + TLP length crosses an AXI Master_Page_Boundary (see Figure 48-40)

The AXI *Master_Page_Boundary* specifies an address page boundary of 4K. The AXI bridge will ensure that the inbound request will not cross the selected page boundary when driven onto AXI master interface. .

48.4.4.1.2.1 Reducing Inbound Decomposition

To reduce decomposition on the inbound path, observe the following guidelines. For Read Requests:

- Ensure that the remote link partner does not issue a read request TLP with a requested length greater than the AXI *Master_Max_Read_Request Size* (see Figure 48-40).

or

- Design your application (and configure the bridge) to handle longer AXI bursts. That is, increase the value of the configuration parameter CC_MSTR_BURST_LEN or increase the AXI master data bus width.

For Write Requests:

- Ensure that the remote link partner does not issue a write request TLP with a payload greater than the AXI *Master_MTU* (see Figure 48-40).

or

- Design your application (and configure the bridge) to handle longer AXI bursts. That is, increase the value of the configuration parameter CC_MSTR_BURST_LEN or increase the AXI master data bus width.

For All Requests:

- Program or design your PCI Express system so that a TLP address plus the TLP length does not cross the AXI *Master_Page_Boundary* (see Figure 48-40).

.

48.5 App Note: Order Enforcement Using the PCIe Core

48.5.1 PCIe Ordering Rule Overview

This application note is written for applications that require certain ordering rules within PCIe traffic. There are many ordering rules according to the PCIe specification. The following is a general description from the PCIe 2.1. base spec.

PCIe 2.1 base specification defines the ordering requirements for PCI Express transactions. The ordering rules defined in the spec apply within a single traffic class (TC). There is no ordering requirement among transactions with different TC levels.

Root Complexes that support peer-to-peer operation and Switches must enforce these transaction ordering rules for all forwarded traffic. These forwarding devices should not forward traffic from one virtual channel to another.

Basic ordering rules are summarized as follows:

- Posted is permitted to pass posted transaction only if the relaxed order bit is set.
- Non-posted is permitted to pass or to be blocked by non-posted.
- Completion with different ID is permitted to pass or be blocked by other completions.
- Posted must be allowed to pass non-posted to avoid deadlock.
- Posted is permitted to be blocked by completions.
- Non-posted can never pass posted.
- Non-posted is permitted to pass or be blocked by completions.
- Completions can pass posted only if the relaxed order bit is set.
- Completions must be allowed to pass non-posted to avoid deadlock.

In general, ordering rules are used based on a specific device's functionality. Root Complexes with peer-to- peer support and switch devices must enforce the above rules. For endpoint devices, it is device-specific. For example, it is possible that an endpoint device is allowed to have completions passing posted and non- posted requests. Because most of the above rules are considered as "permitted or blocked", it is legal to have "permitted" or "blocked" be determined by the architecture of the PCIe core and AXI bridge. The DWC PCIe core and AXI bridge IP is designed such that the order rules may be enforced strictly or less strictly, based on the requirements of the application.

48.5.2 PCIe Core Inbound Order Enforcement

There are three receive queue architectures within the DWC PCIe core. These queue structures will determine different order enforcement based on different receive queue architectures.

If the core is configured with all transactions in bypass mode, then order enforcement should be performed by the application logic. The core will not be responsible for implementing PCIe order enforcement. Since it is an all bypass configuration, all transactions are presented onto the application interface in the order that they arrived.

Below are descriptions for different queueing architectures that are implemented in the PCIe IP core. Three different queue modes are configurable and are designed to suit different applications.

48.5.2.1 Single queue

In Single queue mode, a single header and data queue are populated in the DWC PCIe core for all transactions received. Completions can be bypassed if desired, based on the application. Single queue mode implements strong ordering (in order delivery). All transactions of the same or different traffic class are enqueued and dequeued in the order the transactions were received. For example, if a non-posted transaction is followed by a posted transaction, then the application interface (the DWC PCIe core's target1 interface) will have a non-posted transaction being presented, followed by a posted transaction. Even if the non-posted transaction is blocked by the application (halt), the posted transaction will not be allowed to bypass.

This queue mode has followed all ordering rules above except for rule #4 and rule #9 if completion is not in bypass. What this queuing mode implements is that posted transactions will be blocked by another posted, non-posted will be blocked by another non-posted, etc. That is, we have selected ìblockî as the default of the above ordering rules. According to the spec., these are permitted except for rule #4, and possibly rule #9, if the completion is not in bypass. Rule #4 and rule #9 exist to avoid deadlocks caused by credit starvation and are applicable only when the TLP will be forwarded to another PCI/ PCIe link. Because of these violations, this queue mode is not appropriate for a Root Complex (with peer-to-peer support) or for switch designs. This is designed for endpoint devices that have no order rule requirement at the transaction forwarding between the DWC PCIe core receive queue and the core application interface. The advantage of this queue mode is that it is simple, resulting in an area and power advantage.

48.5.2.2 Multiple queue

In Multiple queue mode, a header and data queue per type of transaction are populated in the DWC PCIe core. All transactions are enqueued in-order. All posted are enqueued into posted header/data queue. All non-posted are enqueued into a header/data non-posted queue and all completions are enqueued into a header/data completion queue. In multiple VC cases, each VC will have its own set of queues.

This queue mode has the same ordering as for single-queue mode. The user has further flexibility in setting which TLPs are bypassed and which TLPs are added to the ordering queue, but these additional modes offer little practical value in the normal case. Since it shares the same limitations as singe-queue mode, this queue mode is not likely appropriate for a Root Complex with peer-to-peer support or for switch designs This queue has some gate count advantage over the full order support provided by the segmented buffer; however, this advantage is generally minor, especially for multi-VC systems.

48.5.2.3 Segmented buffer queue

A single header/data queue are populated in the DWC PCIe core for this queue mode. Transactions of different types are queued into a segmented buffer; each TLP type into a separate segment in the buffer. The transaction's order information is also queued into the segmented buffer. All posted requests are enqueued into the posted segment, all non-posted requests are queued into the non-posted segment and all completions are enqueued into the completion segment. Each traffic class also has its own dedicated per- type segments.

All PCIe order enforcement rules are strictly followed. All transactions are normally served in the order of reception, if the application has no blocking (halt) of the transactions received. If the application does have blocking conditions , then the order rules are enforced for transactions being read out of the queue. For example, if a non-posted request is received followed by a posted request, and the application could not take any more non-posted requests, then the posted will be allowed to pass the non-posted request and the posted transaction will be presented onto the application interface. In a second example, a posted request is received followed by a non-posted request, and the application can not take any more posted requests, then the non-posted transaction will be blocked even if the application can take the next non-posted transaction. Only when there are no previously received posted requests will the non-posted transaction be presented onto the application interface.

Since the order information is stored along with each transaction, full PCIe order enforcement can be accomplished through this mode of operation. This is the difference from the single and multiple queue architectures.

Segmented buffer queue mode is designed for Switches and Root Complexes with peer-to-peer support. The primary advantage of this mode is the full support of PCIe ordering rules. In addition, this mode offers a reduced RAM requirement when compared with multi-queue mode, especially for multi-VC designs.

The figure below represents the various queue architectures of the DWC PCIe IP core inbound receiving traffic. There are three different queue modes. As described above, the ordering rules are implemented differently, based on different queue architecture.

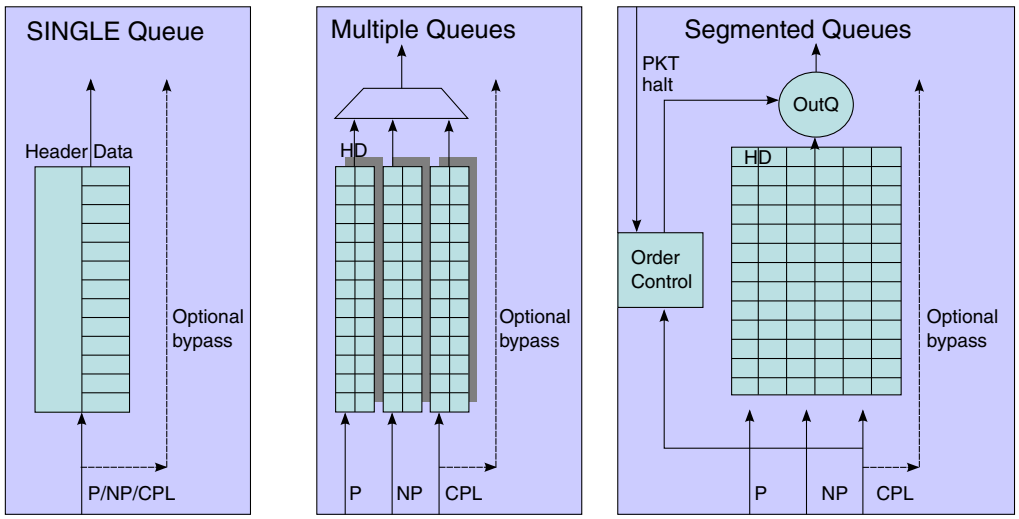


Figure 48-41. DWC PCIe Core Inbound Traffic Queue Architectures

48.5.3 PCIe Core Outbound Order Enforcement

The PCIe Core has a basic transmit architecture such that all transactions are presented outbound onto the PCIe wire based on the order the application presents them onto the PCIe core transmit interfaces.

There are up to three application outbound transmit interfaces (client0, client1 and optionally, client2). All client interfaces are served default as round robin arbitration if credit is available, regardless of the type of transaction. For example, if a posted transaction is presented onto client1 followed by a completion transaction, and if credits permit, then the posted transaction will be transmitted onto the wire before the completion. If the credit is not available, then the completion can pass the posted and be sent onto the wire.

It is the responsibility of the application to make appropriate use of the three client interfaces. An application that has three independent threads of traffic can use the three client interfaces such that the PCIe core will do the arbitration for the outbound transmission. For example, if the application has one source of outbound read transactions and one source of outbound write transactions, and they are independent sources, then client0 and client1 interfaces of the PCIe core should be used to perform the outbound transmission of the transactions. If the read and write are related, then one single interface client0 should be used and it is up to the application to maintain the desired order.

Most applications have independent threads of inbound reads such that its completion is not related to the outbound reads and writes. With these, an independent application interface such as client2 should be used for outbound completions.

In general, there is no transmit queuing in the PCIe core. Therefore, there is no order enforcement among outbound transactions within the transmit queue. The PCIe core has a cut-through transmit architecture where all outbound transaction order rules are enforced by the application. The PCIe core transmits TLPs in the order they are accepted by the core.

Endpoint devices can use two or three client interfaces for posted, non-posted and completion outbound traffic. The two or three client interfaces are determined by whether there is a dependency between the posted and non-posted traffic.

Root Complex and Switch devices will likely use one client interface where the order enforcement is performed by the application logic, where the outbound transfer queue is near to the transmit interface. The credit information is an output of the PCIe core, where it can be used as an input for taking outbound transactions out of the application's queue.

The figure below shows the outbound traffic queue architecture of the PCIe core. As shown in the figure, there is no transmit queue implemented in the core. There is just a retry buffer for replaying PCIe traffic as required by the PCIe base specification. Therefore, outbound TLPs will pass directly from the client interface through the core and to the wire in the same order they were accepted from the application.

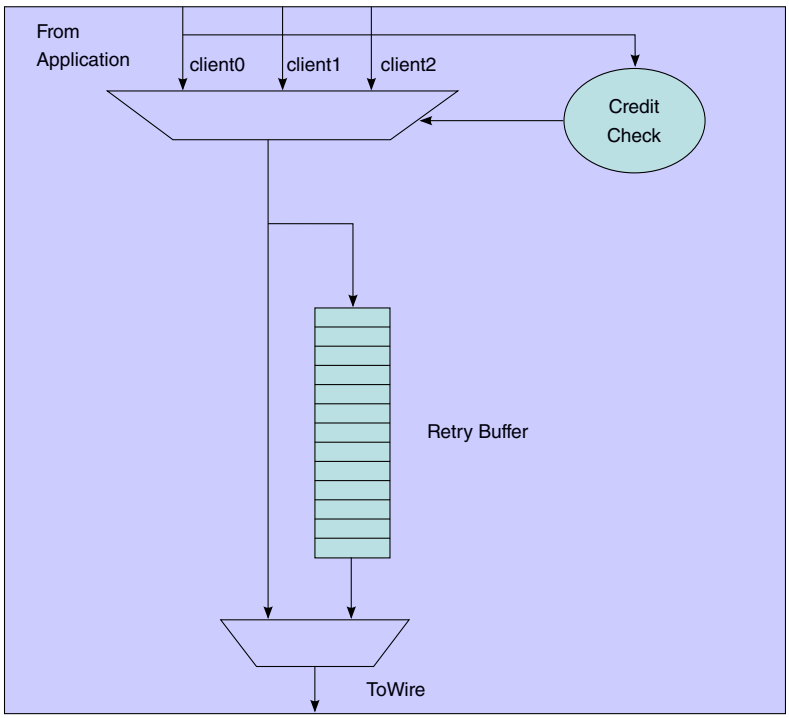


Figure 48-42. PCIe Core Outbound Traffic Queue Architectures

48.5.4 PCIe AHB/AXI Bridge Order Enforcement

Order enforcement through the AXI/AHB bridges is discussed in the following sections:

- Inbound Order Enforcement for AXI Bridge
- Additional Information (ordering)

48.5.5 Additional Information

Additional information is available at:

- Queuing Architecture.
- Queue Modes.
- Order Enforcement.

48.6 App Note: Calculating Gen1 PCI Express and AXI Bridge Throughput

This application note defines the throughput calculation (primarily) with respect to the PCI Express core in Gen1 2.5 GT/s mode.

NOTE

The max transfer size is 128bits

48.6.1 PCI Express Throughput

PCI Express bandwidth is 2.5 Gb/s (gigabits per second), per lane, when operating at the Gen 1 data rate. Because data is encoded using 8b10b encoding, the effective maximum throughput is 250 MB/s (megabytes per second), per lane (and overall throughput since we have 1 lane) , calculated as follows:

2.5 Gb * 8b/10b = 2 Gb * 1B/8b = 250 MB/sec per lane

48.6.2 Effective Throughput

The effective throughput is the payload throughput once all PCIe protocol's overheads have been factored out. The key protocol features are:

- 8b10b encoding at the physical layer. This takes away 20% of the raw bandwidth.
- Acknowledge and flow control update packets at the data link layer (DLLPs). This takes away 1% to 5% of the remaining bandwidth.
- Packet overhead at the transaction layer. Your design choices have a great effect here:
  - Small packets may take away 75% of the bandwidth!
  - Large packets may take away as little as 1%.

48.6.2.1 Effective Throughput Calculation

The table below identifies the TLP package.

Table 48-32. TLP Packet (with associated Data Link Layer overhead bytes)

STP 1 Byte	SEQ 2 Bytes	TLP Header 12/16 Bytes	Data Payload	ECRC 4 Bytes	LCRC 4 Bytes	END 1 Byte

48.6.2.1.1 Packet Level: (Start and End, Link CRC, Header)

The table below summarizes throughput calculations based on a payload size from 16 to 4K bytes.

Table 48-33. Effective Throughput

	Payload Bytes	Header Bytes	ECRC Bytes	PHY and Data Link Layer Bytes	Calculation	Percent Throughput	Note
Worst Packet	16	16	4	8	16/44	36%	Has ECRC.
Typical Packet	128	16	0	8	128/152	84%	No ECRC.
Typical Packet	256	16	0	8	256/280	91%	No ECRC.
Typical Packet	512	16	0	8	512/536	96%	No ECRC.
Best Packet	4096	12	0	8	4096/4116	99%	No ECRC.

NOTE

A 128-byte payload size yields about 67% of the net throughput. Increasing the payload size to 512 bytes increases the net throughput to 92%. Increasing the payload size from 512 bytes to 4096 bytes only contributes an increase of 8% in the net throughput, and the storage requirements are more than doubled. In addition, a large payload size may have an impact on performance due to re- transmission of TLPs. Therefore, 256-byte and 512-byte payload sizes are the most popular choices. Most chipsets support 128 -byte or 256-byte payload sizes, with 512 bytes gaining in popularity.

48.6.2.1.2 Link Layer: (Flow Control and ACK/NAK DLLPs)

DLLPs should be sent as often as required to avoid a negative impact on the TLP throughput. The core sends a pending DLLP if there is no competing TLP traffic.

Sending ACK/NAK and Update FC is controlled by timers. The core provides flexibility to fine-tune these as required. The default setup is minimal flow control (one per time-out) and minimal ACK/NAK device latencies. The link partners retry buffer (payload) size may require a change in the ACK/NAK/Update FC time-out to obtain optimal system latencies.

The core provides a feature to accumulate up to 255 ACKs before issuing an ACK. This feature offers-fine tuning of ACK frequency impact. The core transmits ACKs at fixed intervals, by default.

The following table identifies the DLLP package.

Table 48-34. DLLP Package

STP	Type	Data	16b CRC	END
1 Byte	1 Byte	4 Bytes	2 Bytes	1 Byte

The following table summarizes the throughput calculations.

Table 48-35. Effective Throughput

	Bytes	Result	Percent Throughput
Worst Packet	One ACK plus one FC per 128 (8 packets of 16) bytes of data => 2 DLLPs per 8 data packets	(8*44) / (8*44 + 2*8)	95%
Typical Packet	One ACK plus one FC per 1.4 (256 byte payload) bytes of data => 2 DLLPs per 1.4 packets	1.4*280 / (1.4*280 + 2*8)	96%
Best Packet	One ACK plus one FC per 4096 bytes of data => 2 DLLP per data packet	4116 / (4116 + 2*8)	99%

48.6.2.2 Other Factors Impacting Throughput

Replay buffer and receiver queue sizing should be optimal to avoid a harmful impact of larger ACK/NAK or Update FC latencies by the link partner. The core does a very effective autosizing of the buffers taking into consideration the impact of these parameters.

Effective Throughput (ET) = Raw Throughput * (NL x 2.5 Gb)

Consideration of lane width and payload size results in the following:

Table 48-36. Consideration of lane width and payload size

Factor	Throughput
8b10b	*(80%)
Link Packet	*(95% to 99%)
Data Payload	*(36% to 99%)

The table below identifies lane width and payload vs. throughput.

Table 48-37. Lane Width and Payload vs. Throughput

	Real Throughput Gb/s vs. Data Payload			
Lane Width	16	128	256	4096
x1	0.5	1.7	1.7	2.0

48.7 PCIe CTRL EP Mode Memory Map/Register Definition

PCIE_EP memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1FF_C000	Device ID and Vendor ID Register (PCIE_EP_DeviceID)	32	R	ABCD _16C3h	48.7.1/4198
1FF_C004	Command and Status Register (PCIE_EP_Command)	32	R/W	0000_0000h	48.7.2/4199
1FF_C00C	BIST Register (PCIE_EP_BIST)	32	R/W	0000_0000h	48.7.3/4201
1FF_C010	Base Address 0 (PCIE_EP_BAR0)	32	R	0000_000Ch	48.7.4/4202
1FF_C010	BAR 0 Mask Register (PCIE_EP_MASK0)	32	R	0000_000Ch	48.7.5/4205
1FF_C014	BAR 1 Mask Register (PCIE_EP_MASK1)	32	R	0000_0000h	48.7.6/4207
1FF_C018	BAR 2 Mask Register (PCIE_EP_MASK2)	32	R	0000_0008h	48.7.7/4208
1FF_C01C	BAR 3 Mask Register (PCIE_EP_MASK3)	32	R	0000_0000h	48.7.8/4209
1FF_C028	CardBus CIS Pointer Register (PCIE_EP_CISP)	32	R	0000_0000h	48.7.9/4210
1FF_C02C	Subsystem ID and Subsystem Vendor ID Register (PCIE_EP_SSID)	32	R	0000_0000h	48.7.10/ 4210
1FF_C030	Expansion ROM Base Address Register (PCIE_EP_EROMBAR)	32	R/W	0000_0000h	48.7.11/ 4211
1FF_C030	Expansion ROM BAR Mask Register (PCIE_EP_EROMMASK)	32	R/W	0000_0000h	48.7.12/ 4212
1FF_C034	Capability Pointer Register (PCIE_EP_CAPPR)	32	R	0000_0040h	48.7.13/ 4213
1FF_C03C	Interrupt Line and Pin Register (PCIE_EP_ILR)	32	R/W	0000_01FFh	48.7.14/ 4213
1FF_C100	AER Capability Header (PCIE_EP_AER)	32	R/W	0000_0000h	48.7.15/ 4214
1FF_C104	Uncorrectable Error Status Register (PCIE_EP_UESR)	32	R/W	0000_0000h	48.7.16/ 4215
1FF_C108	Uncorrectable Error Mask Register (PCIE_EP_UEMR)	32	R/W	0000_0000h	48.7.17/ 4217
1FF_C10C	Uncorrectable Error Severity Register (PCIE_EP_UESevR)	32	R/W	000C_2031h	48.7.18/ 4219

Table continues on the next page...

PCIE_EP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1FF_C110	Correctable Error Status Register (PCIE_EP_CESR)	32	R/W	0000_0000h	48.7.19/ 4221
1FF_C114	Correctable Error Mask Register (PCIE_EP_CEMR)	32	R/W	0000_0000h	48.7.20/ 4222
1FF_C118	Advanced Capabilities and Control Register (PCIE_EP_ACCR)	32	R/W	0000_00A0h	48.7.21/ 4224
1FF_C11C	Header Log Register (PCIE_EP_HLR)	32	R	0000_0000h	48.7.22/ 4225
1FF_C140	VC Extended Capability Header (PCIE_EP_VCECHR)	32	R	0000_0012h	48.7.23/ 4225
1FF_C144	Port VC Capability Register 1 (PCIE_EP_PVCCR1)	32	R	0000_0000h	48.7.24/ 4226
1FF_C148	Port VC Capability Register 2 (PCIE_EP_PVCCR2)	32	R	0000_0000h	48.7.25/ 4227
1FF_C14C	Port VC Control and Status Register (PCIE_EP_PVCCSR)	32	R/W	0000_0000h	48.7.26/ 4228
1FF_C150	VC Resource Capability Register n (PCIE_EP_VCRCR)	32	R	0000_0000h	48.7.27/ 4230
1FF_C154	VC Resource Control Register n (PCIE_EP_VCRConR)	32	R/W	0000_0000h	48.7.28/ 4232
1FF_C158	VC Resource Status Register n (PCIE_EP_VCRSR)	32	R	0000_0000h	48.7.29/ 4234

48.7.1 Device ID and Vendor ID Register (PCIE_EP_DeviceID)

Offset: 0x00

The default values of both Device ID and Vendor ID are hardware configuration parameters. The application can overwrite the default values of both Device ID and Vendor ID through the DBI.

Address: 1FF_C000h base + 0h offset = 1FF_C000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																Reserved															
Reset	1	0	1	0	1	0	1	1	1	1	0	0	1	1	0	1	0	0	0	1	0	1	1	0	1	1	0	0	0	0	1	1

PCIE_EP_DeviceID field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
-	This field is reserved. Reserved

48.7.2 Command and Status Register (PCIE_EP_Command)

Offset: 0x04

Bytes: 0-1

Address: 1FF_C000h base + 4h offset = 1FF_C004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Signaled_System_Error	Detected_Parity_Error	Received_Master_Abort	Received_Target_Abort	Signaled_Target_Abort	DEVSEL_Timing		Master_Data_Parity_Error	Fast_Back_to_Back_Capable	Reserved	SixtySix_MHz_Capable	Capabilities_List	INTx_Status	Reserved		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved					INTx_Assertion_Disable	Fast_Back_to_Back_Enable	SERR_Enable	IDSEL_Stepping	Parity_Error_Response	VGA_Palette_Snoop	Memory_Write_and_Invalidate	Special_Cycle_Enable	Bus_Master_Enable	Memory_Space_Enable	I_O_Space_Enable
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_EP_Command field descriptions

Field	Description
31 Signaled_ System_Error	Signaled System Error
30 Detected_Parity_ Error	Detected Parity Error
29 Received_ Master_Abort	Received Master Abort
28 Received_ Target_Abort	Received Target Abort
27 Signaled_Target_ Abort	Signaled Target Abort

Table continues on the next page...

PCIE_EP_Command field descriptions (continued)

Field	Description
26–25 DEVSEL_Timing	DEVSEL Timing Not applicable for PCI Express. Hardwired to 0.
24 Master_Data_Parity_Error	Master Data Parity Error
23 Fast_Back_to_Back_Capable	Fast Back-to-Back Capable Not applicable for PCI Express. Hardwired to 0.
22 -	This field is reserved. Reserved
21 SixtySix_MHz_Capable	66 MHz Capable Not applicable for PCI Express. Hardwired to 0.
20 Capabilities_List	Capabilities List Indicates presence of an extended capability item. Hardwired to 1.
19 INTx_Status	INTx Status
18–16 -	This field is reserved. Reserved
15–11 -	This field is reserved. Reserved
10 INTx_Assertion_Disable	INTx Assertion Disable
9 Fast_Back_to_Back_Enable	Fast Back-to-Back Enable Not applicable for PCI Express. Must be hardwired to 0.
8 SERR_Enable	SERR# Enable
7 IDSEL_Stepping	IDSEL Stepping/Wait Cycle Control Not applicable for PCI Express. Must be hardwired to 0
6 Parity_Error_Response	Parity Error Response
5 VGA_Palette_Snoop	VGA Palette Snoop Not applicable for PCI Express. Must be hardwired to 0.
4 Memory_Write_and_Invalidate	Memory Write and Invalidate Not applicable for PCI Express. Must be hardwired to 0.
3 Special_Cycle_Enable	Special Cycle Enable Not applicable for PCI Express. Must be hardwired to 0.
2 Bus_Master_Enable	Bus Master Enable

Table continues on the next page...

PCIE_EP_Command field descriptions (continued)

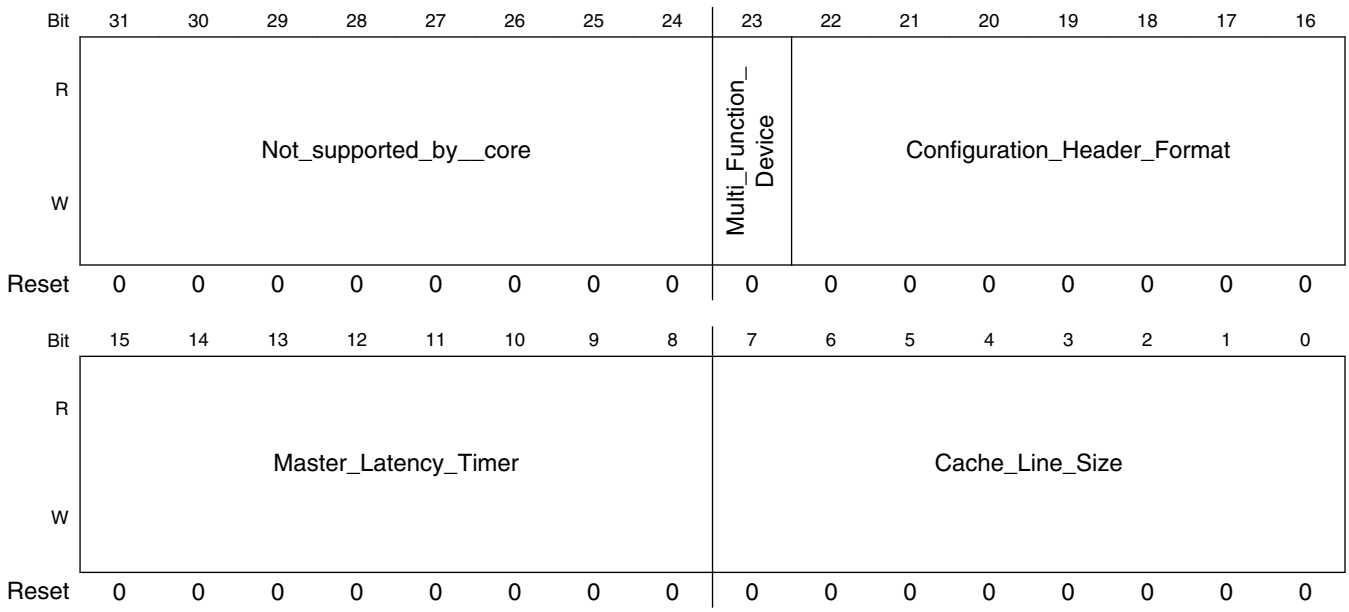
Field	Description
1 Memory_Space_Enable	Memory Space Enable
0 I_O_Space_Enable	I/O Space Enable

48.7.3 BIST Register (PCIE_EP_BIST)

Offset: 0x0C

Byte: 0

Address: 1FF_C000h base + Ch offset = 1FF_C00Ch



PCIE_EP_BIST field descriptions

Field	Description
31–24 Not_supported_by__core	The BIST register functions are not supported by the core. All 8 bits of the BIST register are hardwired to 0.
23 Multi_Function_Device	Multi Function Device The default value is 0 for a single function device (`CX_NFUNC = 1) or 1 for a multi-function device (`CX_NFUNC != 1). The Multi Function Device bit is writable through the DBI.
22–16 Configuration_Header_Format	Configuration Header Format Hardwired to 0 for type 0.

Table continues on the next page...

PCIE_EP_BIST field descriptions (continued)

Field	Description
15–8 Master_Latency_Timer	Master Latency Timer Not applicable for PCI Express, hardwired to 0.
Cache_Line_Size	Cache Line Size The Cache Line Size register is RW for legacy compatibility purposes and is not applicable to PCI Express device functionality. Writing to the Cache Line Size register does not impact functionality of the core.

48.7.4 Base Address 0 (PCIE_EP_BAR0)

Offset: 0x10-0x24

The core provides three pairs of 32-bit BARs for each implemented function. Each pair (BARs 0 and 1, BARs 2 and 3, BARs 4 and 5) can be configured as follows:

- One 64-bit BAR: For example, BARs 0 and 1 are combined to form a single 64-bit BAR.
- Two 32-bit BARs: For example, BARs 0 and 1 are two independent 32-bit BARs.
- One 32-bit BAR: For example, BAR 0 is a 32-bit BAR and BAR 1 is either disabled or removed from the core altogether to reduce gate count.

In addition, you can configure each BAR to have its incoming Requests routed to either:

- RTRGT1
-

The following sections describe how to set up the BAR types and sizes by programming values into the base address registers. For more information about routing Requests to either RTRGT1 on a BAR-by- BAR basis, see Receive Filtering.

The contents of the six BARs determine the BAR configuration. The reset values of the BARs are determined by hardware configuration options.

At runtime, application software can overwrite the BAR contents to reconfigure the BARs (unless the affected BAR is removed during hardware configuration). Application software must observe the rules listed below when writing to the BARs.

The rules for BAR configuration are the same for all three pairs. Using BARs 0 and 1 as the example pair, the rules for BAR configuration are:

- Any pair (for example, BARs 0 and 1) can be configured as one 64-bit BAR, two 32-bit BARs, or one 32-bit BAR.

- BAR pairs cannot overlap to form a 64-bit BAR. For example, you cannot combine BARs 1 and 2 to form a 64-bit BAR.
-
- An I/O BAR must be a 32-bit BAR and cannot be prefetchable.
- If the device is configured as a PCI Express Endpoint (not a Legacy Endpoint), then any memory that is configured as prefetchable must be a 64-bit memory BAR.
- If BAR 0 is configured as a 64-bit BAR:

- BAR 1 is the upper 32 bits of the combined 64-bit BAR formed by BARs 0 and 1. Therefore, BAR 1 must be disabled and cannot be configured independently.
- BAR 0 must be a memory BAR and can be either prefetchable or non-prefetchable.
- The contents of the BAR 0 Mask register determine the number of writable bits in the 64-bit BAR, subject to the restrictions described in BAR Mask Registers . The BAR 1 Mask register contains the upper 32 bits of the BAR 0 Mask value.
- BAR 0 can be disabled by writing 0 to bit 0 of the BAR 0 Mask register

- If BAR 0 is configured as a 32-bit BAR:

- You can configure BAR 1 as an independent 32-bit BAR
- BAR 0 can be configured as a memory BAR or an I/O BAR.
- The contents of the BAR 0 Mask register determine the number of writable bits in the 32-bit BAR 0, subject to the restrictions described in BAR Mask Registers.
- BAR 0 can be disabled by writing 0 to bit 0 of the BAR 0 Mask register

- When BAR 0 is configured as a 32-bit BAR, BAR 1 is available as an independent 32-bit BAR according to the following rules:

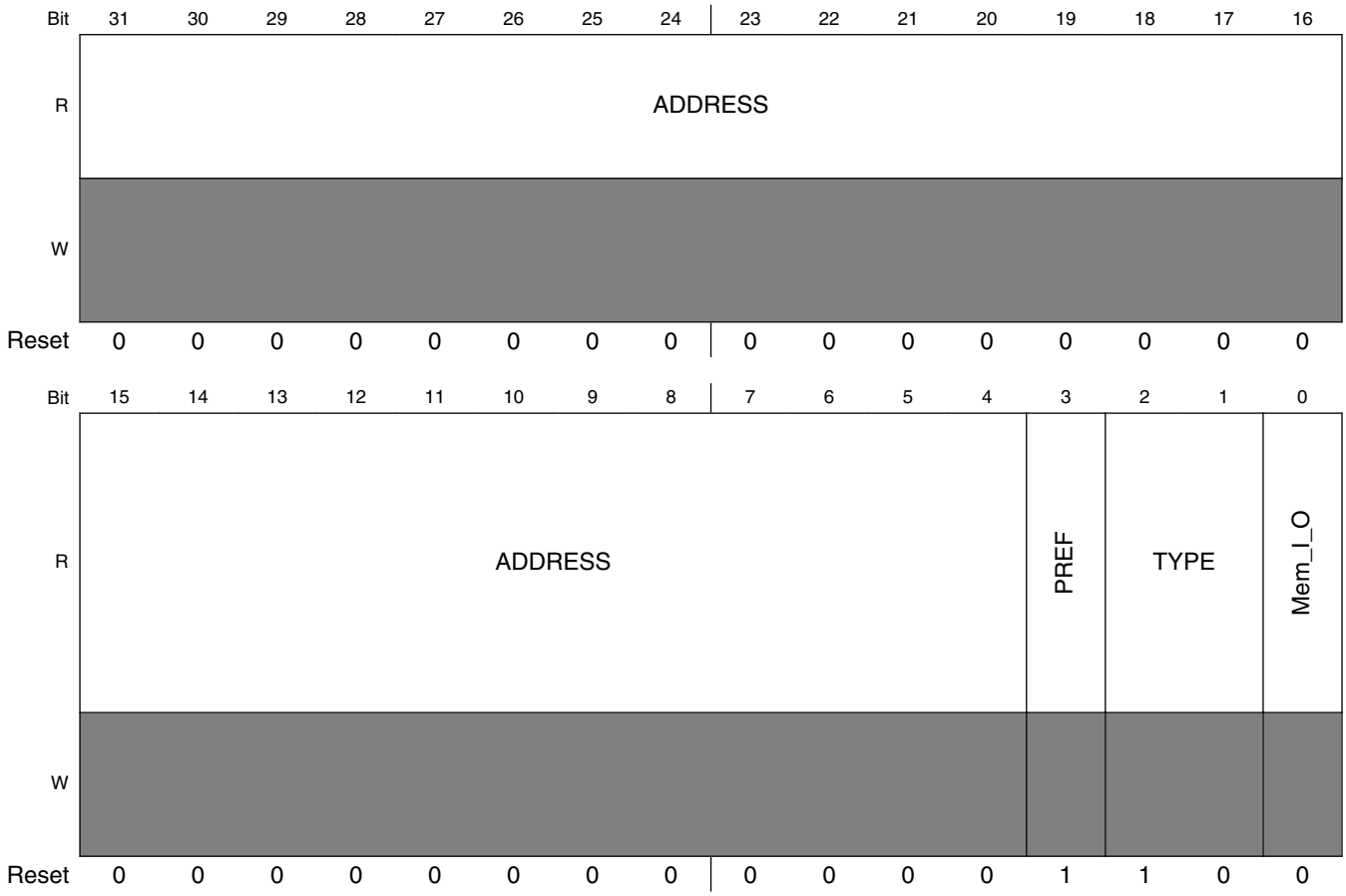
- BAR 1 can be configured as a memory BAR or an I/O BAR.
- The contents of the BAR 1 Mask register determine the number of writable bits in the 32-bit BAR 1, subject to the restrictions described in BAR Mask Registers.
-
-

The same rules apply for pairs 2/3 and 4/5.

Offset: 0x10 (if included in the core hardware configuration)

PCIe CTRL EP Mode Memory Map/Register Definition

Address: 1FF_C000h base + 10h offset = 1FF_C010h



PCIE_EP_BAR0 field descriptions

Field	Description
31–4 ADDRESS	BAR 0 base address bits (for a 64-bit BAR, the remaining upper address bits are in BAR 1). The BAR 0 Mask value determines which address bits are masked.
3 PREF	If BAR 0 is an I/O BAR, bit 3 is the second least significant bit of the base address. Bits [3:0] are writable through the DBI. If BAR 0 is a memory BAR, bit 3 indicates if the memory region is prefetchable: 0 = Non-prefetchable 1 = Prefetchable
2–1 TYPE	If BAR 0 is an I/O BAR, bit 2 the least significant bit of the base address and bit 1 is 0. Bits [3:0] are writable through the DBI. If BAR 0 is a memory BAR, bits [2:1] determine the BAR type: 00 = 32-bit BAR 10 = 64-bit BAR
0 Mem_I_O	Bits [3:0] are writable through the DBI. 0 = BAR 0 is a memory BAR 1 = BAR 0 is an I/O BAR

48.7.5 BAR 0 Mask Register (PCIE_EP_MASK0)

The BAR masks are used for indicating the amount of memory each BAR requests from host software. The application logic can overwrite the default values via the DBI.

The BAR Mask registers determine which bits in each BAR are non-writable by host software, which determines the size of the address space claimed by each BAR.

The BAR Mask values indicate the range of low-order bits in each implemented BAR not to use for address matching. The BAR Mask value also indicates the range of low-order bits in the BAR that cannot be written from the host. The application can write to all BAR bits to allow setting of memory, I/O, and other standard BAR options.

To disable any BAR, the application can write a 0 to bit 0 of the corresponding BAR Mask register. To change the BAR Mask value for a disabled BAR, the application must first enable the BAR by writing 1 to bit 0. After enabling the BAR, the application can then write a new value to the BAR Mask register.

The BAR Mask registers are accessible through the same address as the corresponding BAR registers, but requires dbi_cs2 assertions instead i.e. bit address 12 must be set. The BAR Mask registers are writable only, not readable.

If the BAR Mask value for a BAR is less than that required for the BAR type, the core automatically uses the minimum value for the BAR type:

BAR bits [11:0] are always masked for a memory BAR. The core requires each memory BAR to claim at least 4 KB.

The PCI Express Base Specification states that the minimum memory address range requested by a BAR is 128 bytes. In the PCI Local Bus Specification, Rev 3.0 it is recommended that devices that need less than 4 KB of Address Space should still consume 4 KB of address space in order to minimize the number of bits in the address decoder. A Memory BAR size of 256 bytes can be achieved by using a DBI2 write to BAR Mask.

BAR bits [7:0] are always masked for an I/O BAR. The core requires each I/O BAR to claim at least 256 bytes.

The PCI Local Bus Specification, Rev 3.0 allows I/O BARs to consume between 4 bytes and 256 bytes of address space. The core only permits I/O BARs to consume 256 bytes of address space. This restriction is used in order to minimize the number of bits in the address decoder.

The aperture of the BAR is actually the larger of the written sizeor the system page size (set by the operating system). In the case where the system page size is larger than the requested bar size, the BAR is actually sized to the system page size . This means that when the OS writes all 1s then reads back to determine the size of the BAR, the OS will see the BAR size to be the system page size. The application logic, most likely, will only have the original requested amount of physical memory. A transaction will receive a UR if the transaction is from the RC and it targets an address that is within the range of the allocated system page size but above the implemented application memory.

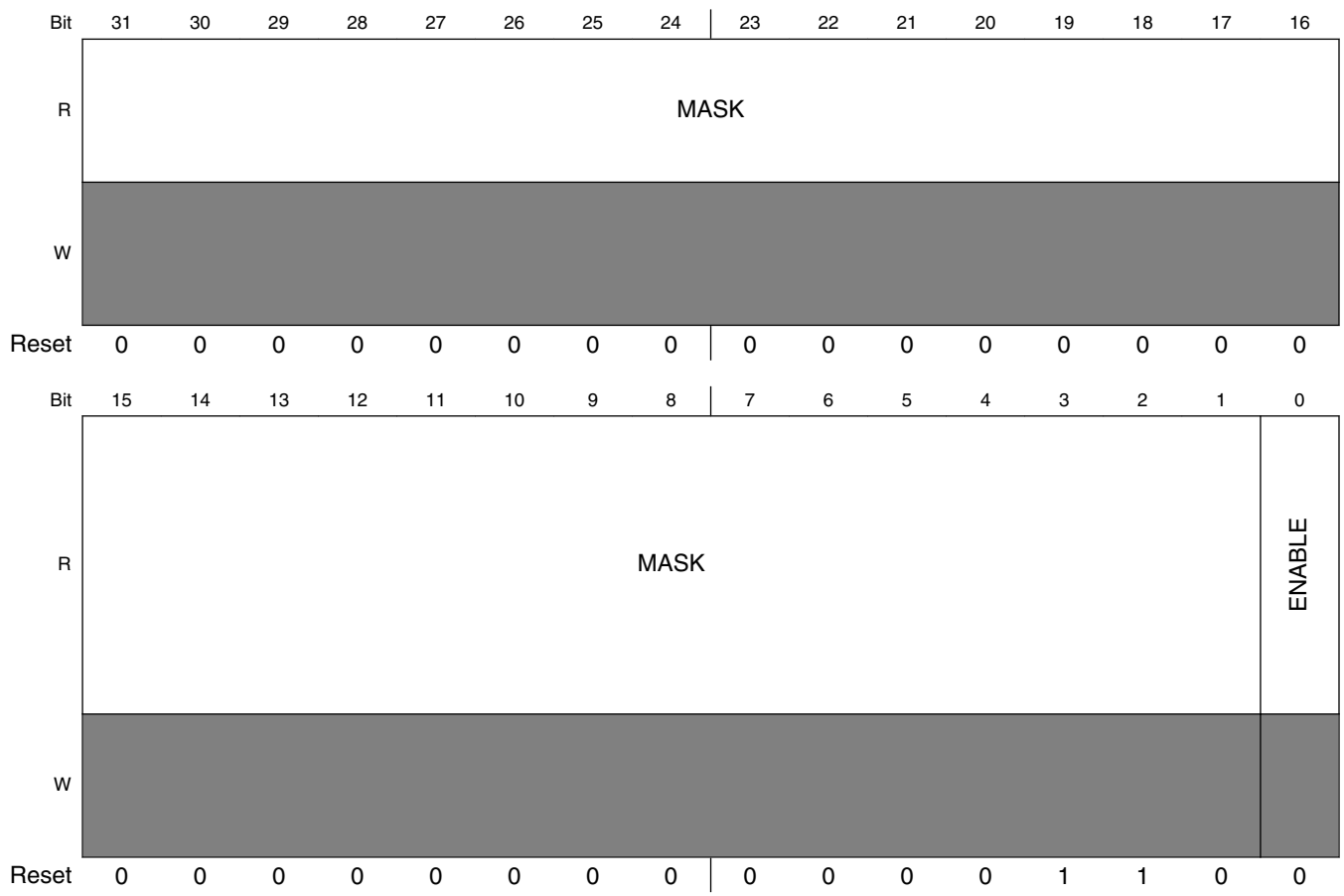
The figure below shows an example configuration of the six BARs and their corresponding BAR Mask registers. The example configuration includes:

- One 64-bit memory BAR (non-prefetchable)
- One 32-bit memory BAR (non-prefetchable)
- One 32-bit I/O BAR

Figure 48-48. Example Base Address Register Configuration

Offset: 0x10 (same as Base Address Register 0, but requires dbi_cs2 for write access)

Address: 1FF_C000h base + 10h offset = 1FF_C010h



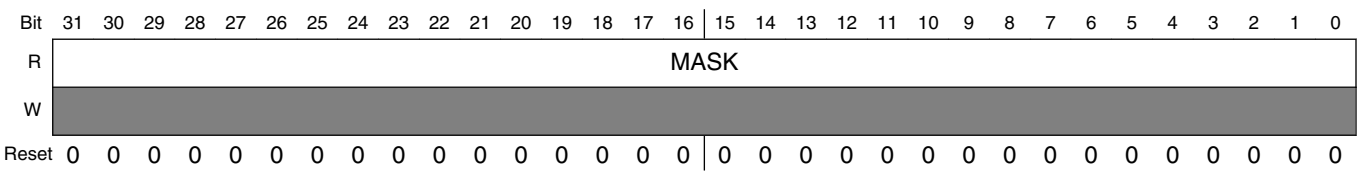
PCIE_EP_MASK0 field descriptions

Field	Description
31–1 MASK	Indicates which BAR 0 bits to mask (make non- writable) from host software, which, in turn, determines the size of the BAR. For example, writing 0xFFF to the BAR 0 Mask register claims a 4096- byte BAR by masking bits 11:0 of the BAR from writing by host software. The BAR 1 Mask register contains the upper bits of the BAR 0 Mask. The BAR 0 Mask register is invisible to host software and not readable from the application. • the BAR 0 Mask register is writable through the DBI. • •
0 ENABLE	Bit 0 is interpreted as BAR Enable when writing to the BAR Mask register rather than as a mask bit because bit 0 of a BAR is always masked from writing by host software. BAR 0 Enable 0 BAR 0 is disabled 1 BAR 0 is enabled

48.7.6 BAR 1 Mask Register (PCIE_EP_MASK1)

Offset: 0x14 (same as Base Address Register 1, but requires dbi_cs2 for write access)

Address: 1FF_C000h base + 14h offset = 1FF_C014h



PCIE_EP_MASK1 field descriptions

Field	Description
MASK	• Bits [31:1]: BAR 1 Mask value, interpreted the same way as BAR 0 Mask. Default value is `BAR1_MASK_N. • Bit 0: BAR 1 Enable (0 = BAR 1 is disabled; 1= BAR 1 is enabled). Default value is `BAR1_ENABLED_N. • `BAR1_MASK_WRITABLE_N controls application write access to the BAR 1 Mask register. • Bits [31:0] are the upper bits of the BAR 0 Mask value. • `BAR0_MASK_WRITABLE_N controls application write access to the full 64-bit BAR 0 Mask register.

48.7.7 BAR 2 Mask Register (PCIE_EP_MASK2)

Offset: 0x18 (same as Base Address Register 2, but requires dbi_cs2 for write access)

Address: 1FF_C000h base + 18h offset = 1FF_C018h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BAR2_MASK_N															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BAR2_MASK_N															BAR2_ENABLED_N
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

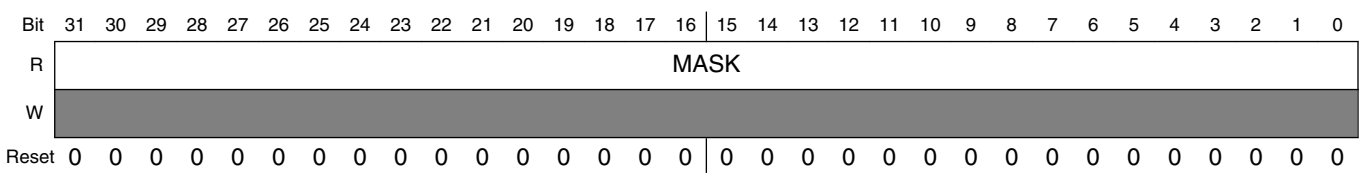
PCIE_EP_MASK2 field descriptions

Field	Description
31–1 BAR2_MASK_N	Indicates which BAR 2 bits to mask (make non- writable) from host software, which, in turn, determines the size of the BAR. For example, writing 0xFFF to the BAR 2 Mask register claims a 4096-byte BAR by masking bits 11:0 of the BAR from writing by host software. The BAR 2 Mask register is invisible to host software and not readable from the application. • the BAR 2 Mask register is writable through the DBI. •
0 BAR2_ ENABLED_N	Bit 0 is interpreted as BAR Enable when writing to the BAR Mask register rather than as a mask bit because bit 0 of a BAR is always masked from writing by host software. BAR 2 Enable 0 BAR 2 is disabled 1 BAR 2 is enabled

48.7.8 BAR 3 Mask Register (PCIE_EP_MASK3)

Offset: 0x1C (same as Base Address Register 3, but requires dbi_cs2 for write access)

Address: 1FF_C000h base + 1Ch offset = 1FF_C01Ch



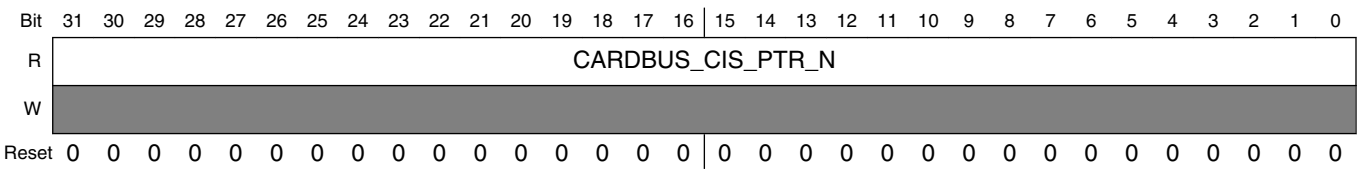
PCIE_EP_MASK3 field descriptions

Field	Description
MASK	• Bits [31:1]: BAR 3 Mask value, interpreted the same way as BAR 2 Mask. Default value is `BAR3_MASK_N. • Bit 0: BAR 3 Enable (0 = BAR 3 is disabled; 1= BAR 3 is enabled). Default value is `BAR3_ENABLED_N. • `BAR3_MASK_WRITABLE_N controls application can not write access to the BAR 3 Mask register. • Bits [31:0] are the upper bits of the BAR 2 Mask value. • `BAR2_MASK_WRITABLE_N controls application write access to the full 64-bit BAR 2 Mask register.

48.7.9 CardBus CIS Pointer Register (PCIE_EP_CISP)

Offset: 0x28

Address: 1FF_C000h base + 28h offset = 1FF_C028h



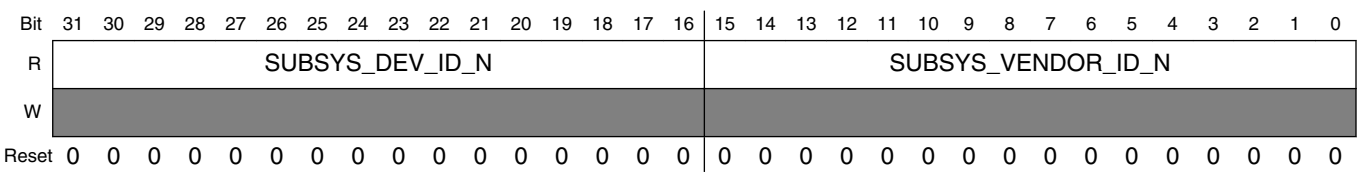
PCIE_EP_CISP field descriptions

Field	Description
CARDBUS_CIS_PTR_N	CardBus CIS Pointer Optional, writable through the DBI.

48.7.10 Subsystem ID and Subsystem Vendor ID Register (PCIE_EP_SSID)

Offset: 0x2C

Address: 1FF_C000h base + 2Ch offset = 1FF_C02Ch



PCIE_EP_SSID field descriptions

Field	Description
31–16 SUBSYS_DEV_ID_N	Subsystem ID Writable through the DBI.
SUBSYS_VENDOR_ID_N	Subsystem Vendor ID Writable through the DBI.

48.7.11 Expansion ROM Base Address Register (PCIE_EP_EROMBAR)

Offset: 0x30

Address: 1FF_C000h base + 30h offset = 1FF_C030h

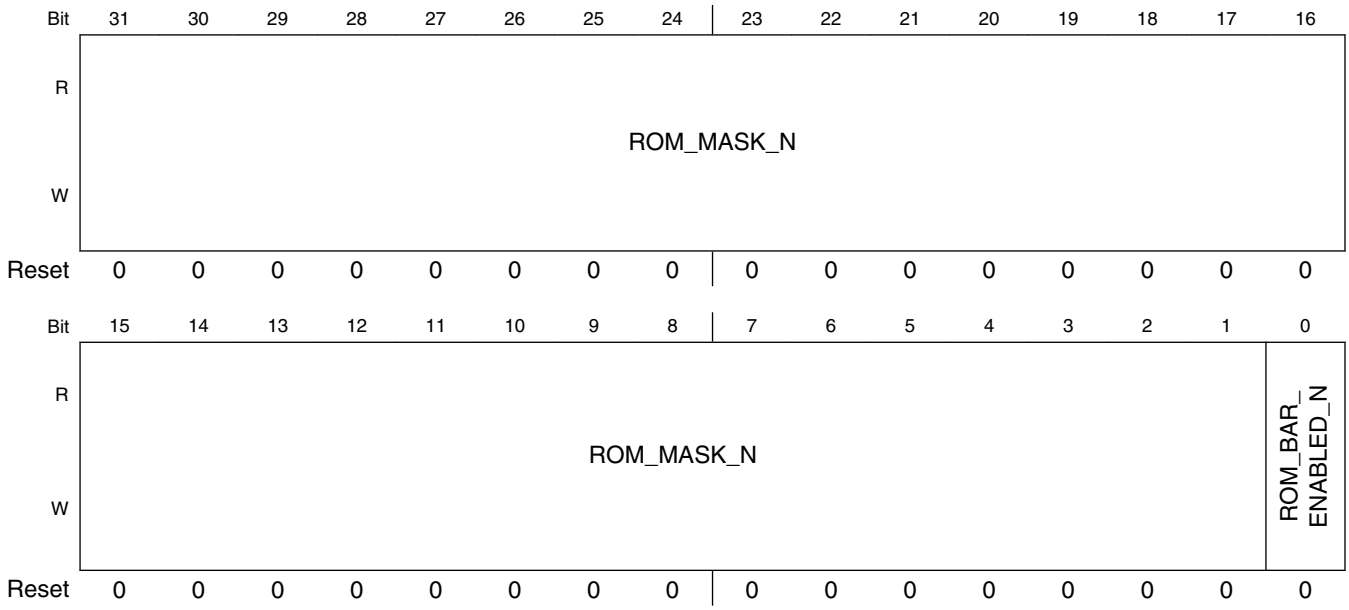
PCIE_EP_EROMBAR field descriptions

Field	Description
31–11 ADDRESS	Expansion ROM Address
10–1 -	This field is reserved. Reserved
0 ENABLE	Expansion ROM Enable

48.7.12 Expansion ROM BAR Mask Register (PCIE_EP_EROMMASK)

Offset: 0x30 (same as the Expansion ROM BAR, but requires dbi_cs2 for write access)

Address: 1FF_C000h base + 30h offset = 1FF_C030h



PCIE_EP_EROMMASK field descriptions

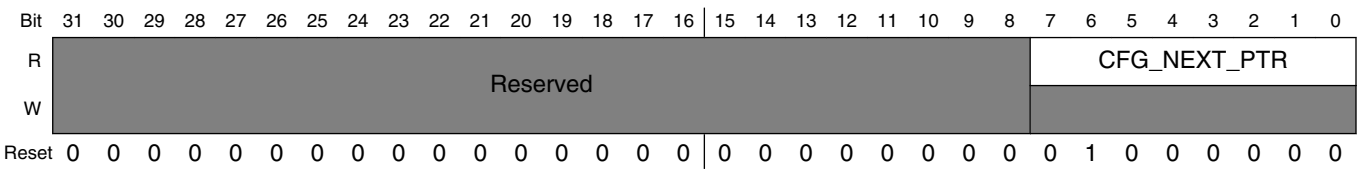
Field	Description
31–1 ROM_MASK_N	Indicates which Expansion ROM BAR bits to mask (make non-writable) from host software, which, in turn, determines the size of the BAR. For example, writing 0xFFF to the Expansion ROM BAR Mask register claims a 4096-byte BAR by masking bits 11:0 of the BAR from writing by host software. The maximum value is 0xFFFFFF because the maximum space that can be claimed by an Expansion ROM BAR is 16 MB. The Expansion ROM BAR Mask register is invisible to host software and not readable from the application. Application access depends on the value of • the Expansion ROM BAR Mask register is writable through the DBI. •
0 ROM_BAR_ ENABLED_N	Expansion ROM BAR Enable 0 Expansion ROM BAR is disabled 1 Expansion ROM BAR is enabled

48.7.13 Capability Pointer Register (PCIE_EP_CAPPR)

Offset: 0x34

Byte: 0

Address: 1FF_C000h base + 34h offset = 1FF_C034h



PCIE_EP_CAPPR field descriptions

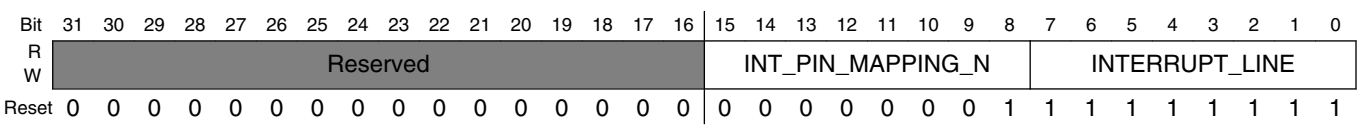
Field	Description
31–8 -	This field is reserved. Reserved
CFG_NEXT_PTR	First Capability Pointer. See PF PCI Standard Capability Structures Register Maps for more information.

48.7.14 Interrupt Line and Pin Register (PCIE_EP_ILR)

Offset: 0x3C

Byte: 0

Address: 1FF_C000h base + 3Ch offset = 1FF_C03Ch



PCIE_EP_ILR field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
15–8 INT_PIN_ MAPPING_N	Interrupt Pin Identifies the legacy interrupt Message that the device (or device function) uses. In a single-function configuration, the core only uses INTA. The Interrupt Pin register is writable through the DBI. Valid values are: 0x00 The device (or function) does not use legacy interrupt

Table continues on the next page...

PCIE_EP_ILR field descriptions (continued)

Field	Description
	0x01 The device (or function) uses INTA 0x02 The device (or function) uses INTB 0x03 The device (or function) uses INTC 0x04 The device (or function) uses INTD
INTERRUPT_ LINE	Interrupt Line Value in this register is system architecture specific. POST software will write the routing information into this register as it initializes and configures the system.

48.7.15 AER Capability Header (PCIE_EP_AER)

The core implements the following PCI Express Extended Capabilities registers:

? Advanced Error Reporting Capability register set

? Virtual Channel Capability register set -

Address: 0x100

Address: 1FF_C000h base + 100h offset = 1FF_C100h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Next_Capability_Offset												Capability_ Version				PCI_Express_Extended_Capability_ID															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

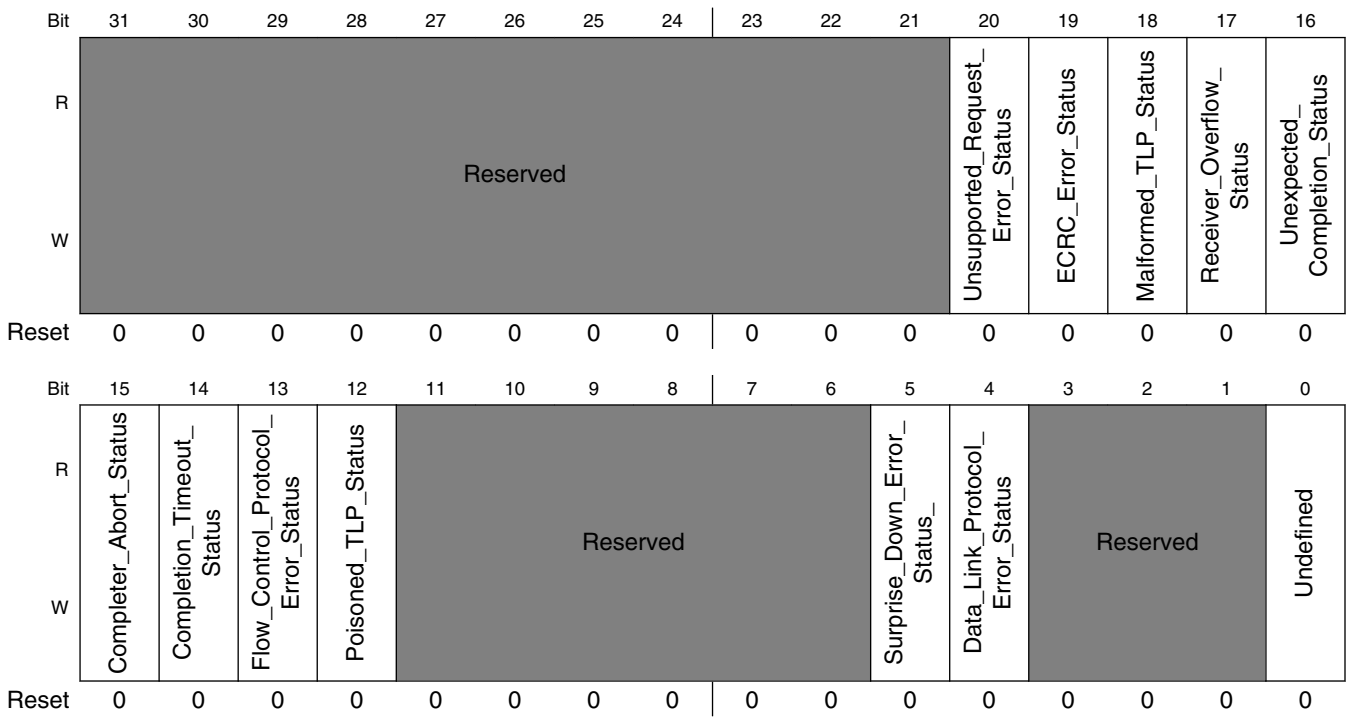
PCIE_EP_AER field descriptions

Field	Description
31–20 Next_Capability_ Offset	Next Capability Offset
19–16 Capability_ Version	Capability Version
PCI_Express_ Extended_ Capability_ID	PCI Express Extended Capability ID Value is 0x1 for Advanced Error Reporting.

48.7.16 Uncorrectable Error Status Register (PCIE_EP_UESR)

Offset: 0x04

Address: 1FF_C000h base + 104h offset = 1FF_C104h



PCIE_EP_UESR field descriptions

Field	Description
31–21 -	This field is reserved. Reserved
20 Unsupported_Request_Error_Status	Unsupported Request Error Status
19 ECRC_Error_Status	ECRC Error Status
18 Malformed_TLP_Status	Malformed TLP Status
17 Receiver_Overflow_Status	Receiver Overflow Status

Table continues on the next page...

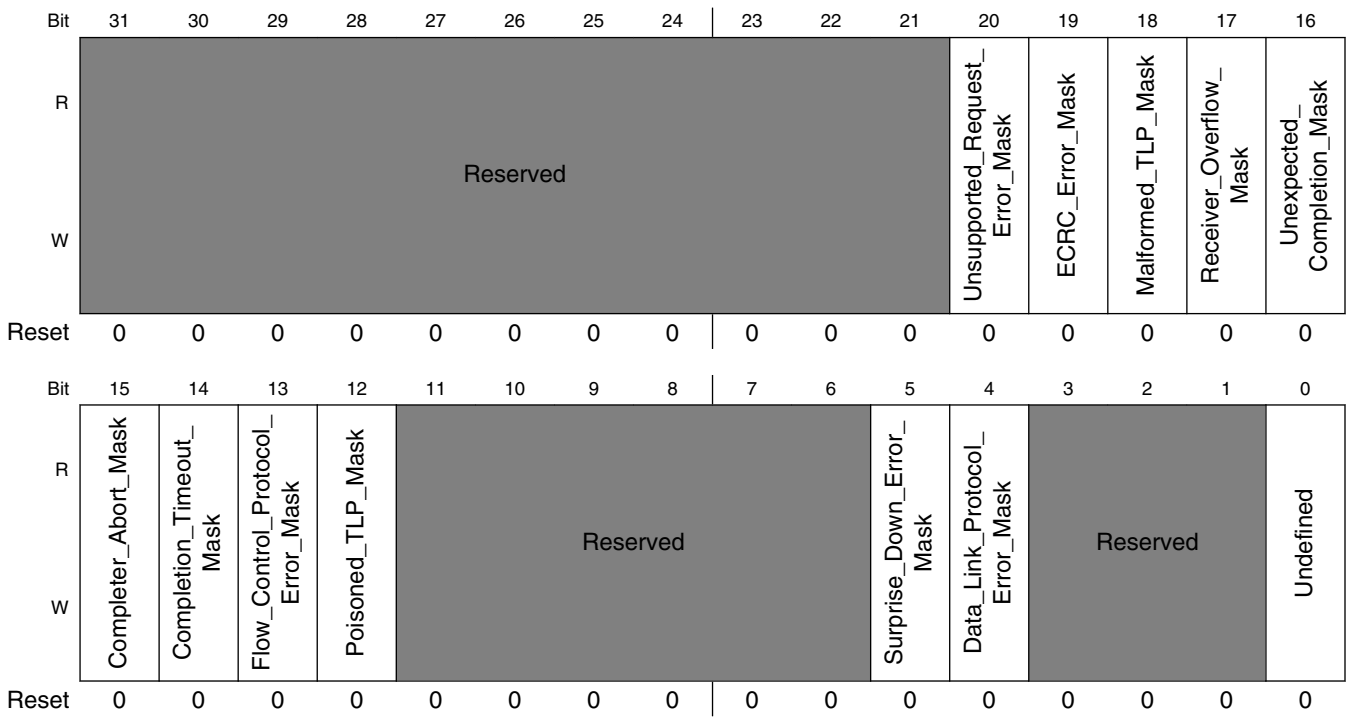
PCIE_EP_UESR field descriptions (continued)

Field	Description
16 Unexpected_ Completion_ Status	Unexpected Completion Status
15 Completer_ Abort_Status	Completer Abort Status
14 Completion_ Timeout_Status	Completion Timeout Status
13 Flow_Control_ Protocol_Error_ Status	Flow Control Protocol Error Status
12 Poisoned_TLP_ Status	Poisoned TLP Status
11–6 -	This field is reserved. Reserved
5 Surprise_Down_ Error_Status_	Surprise Down Error Status (not supported)
4 Data_Link_ Protocol_Error_ Status	Data Link Protocol Error Status
3–1 -	This field is reserved. Reserved
0 Undefined	Undefined for PCI Express 1.1 (Was Training Error Status for PCI Express 1.0a)

48.7.17 Uncorrectable Error Mask Register (PCIE_EP_UEMR)

Offset: 0x08

Address: 1FF_C000h base + 108h offset = 1FF_C108h



PCIE_EP_UEMR field descriptions

Field	Description
31–21 -	This field is reserved. Reserved
20 Unsupported_Request_Error_Mask	Unsupported Request Error Mask
19 ECRC_Error_Mask	ECRC Error Mask
18 Malformed_TLP_Mask	Malformed TLP Mask
17 Receiver_Overflow_Mask	Receiver Overflow Mask

Table continues on the next page...

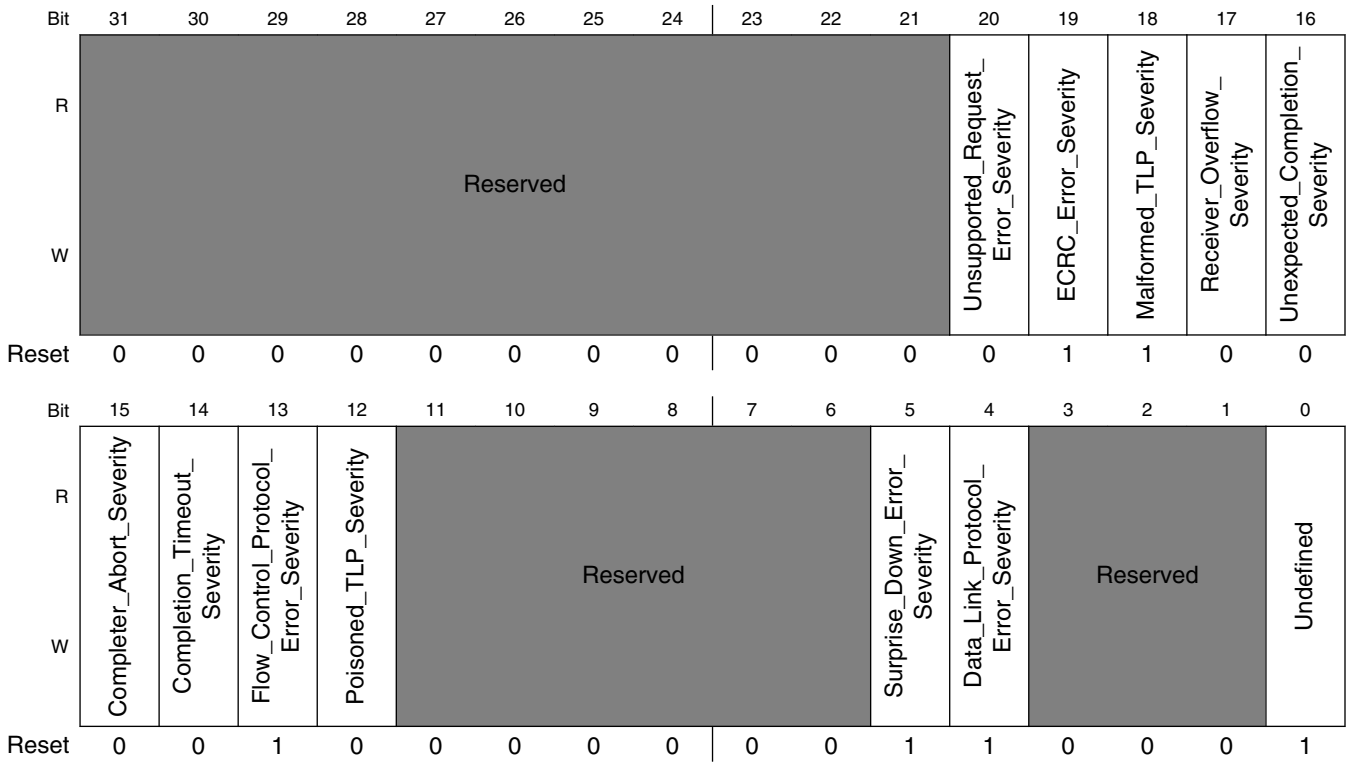
PCIE_EP_UEMR field descriptions (continued)

Field	Description
16 Unexpected_ Completion_ Mask	Unexpected Completion Mask
15 Completer_ Abort_Mask	Completer Abort Mask
14 Completion_ Timeout_Mask	Completion Timeout Mask
13 Flow_Control_ Protocol_Error_ Mask	Flow Control Protocol Error Mask
12 Poisoned_TLP_ Mask	Poisoned TLP Mask
11–6 -	This field is reserved. Reserved
5 Surprise_Down_ Error_Mask	Surprise Down Error Mask (not supported)
4 Data_Link_ Protocol_Error_ Mask	Data Link Protocol Error Mask
3–1 -	This field is reserved. Reserved
0 Undefined	Undefined for PCI Express 1.1 (Was Training Error Mask for PCI Express 1.0a)

48.7.18 Uncorrectable Error Severity Register (PCIE_EP_UESevR)

Offset: 0x0C

Address: 1FF_C000h base + 10Ch offset = 1FF_C10Ch



PCIE_EP_UESevR field descriptions

Field	Description
31–21 -	This field is reserved. Reserved
20 Unsupported_Request_Error_Severity	Unsupported Request Error Severity
19 ECRC_Error_Severity	ECRC Error Severity
18 Malformed_TLP_Severity	Malformed TLP Severity

Table continues on the next page...

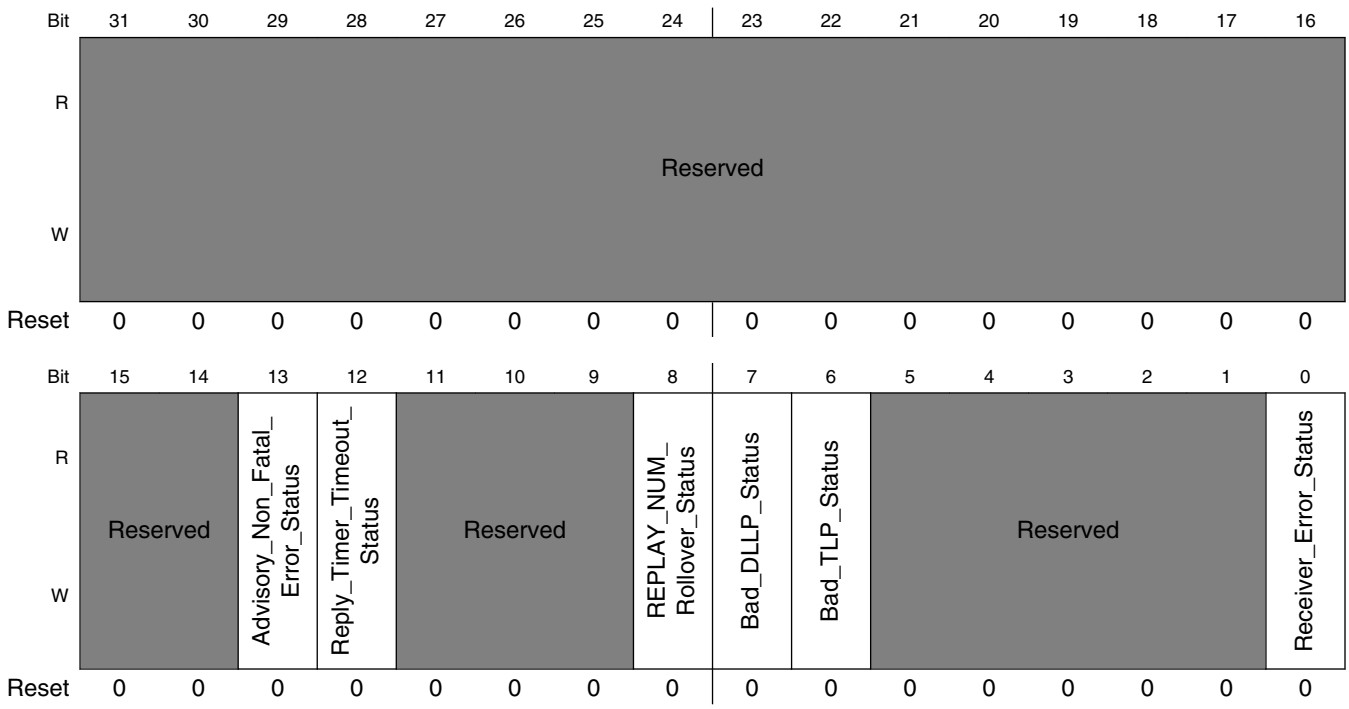
PCIE_EP_UESevR field descriptions (continued)

Field	Description
17 Receiver_ Overflow_ Severity	Receiver Overflow Severity
16 Unexpected_ Completion_ Severity	Unexpected Completion Severity
15 Completer_ Abort_Severity	Completer Abort Severity
14 Completion_ Timeout_Severity	Completion Timeout Severity
13 Flow_Control_ Protocol_Error_ Severity	Flow Control Protocol Error Severity
12 Poisoned_TLP_ Severity	Poisoned TLP Severity
11–6 -	This field is reserved. Reserved
5 Surprise_Down_ Error_Severity	Surprise Down Error Severity (not supported)
4 Data_Link_ Protocol_Error_ Severity	Data Link Protocol Error Severity
3–1 -	This field is reserved. Reserved
0 Undefined	Undefined for PCI Express 1.1 (Was Training Error Severity for PCI Express 1.0a)

48.7.19 Correctable Error Status Register (PCIE_EP_CESR)

Offset: 0x10

Address: 1FF_C000h base + 110h offset = 1FF_C110h



PCIE_EP_CESR field descriptions

Field	Description
31–14 -	This field is reserved. Reserved
13 Advisory_Non_Fatal_Error_Status	Advisory Non-Fatal Error Status
12 Reply_Timer_Timeout_Status	Reply Timer Timeout Status
11–9 -	This field is reserved. Reserved
8 REPLAY_NUM_Rollover_Status	REPLAY_NUM Rollover Status
7 Bad_DLLP_Status	Bad DLLP Status

Table continues on the next page...

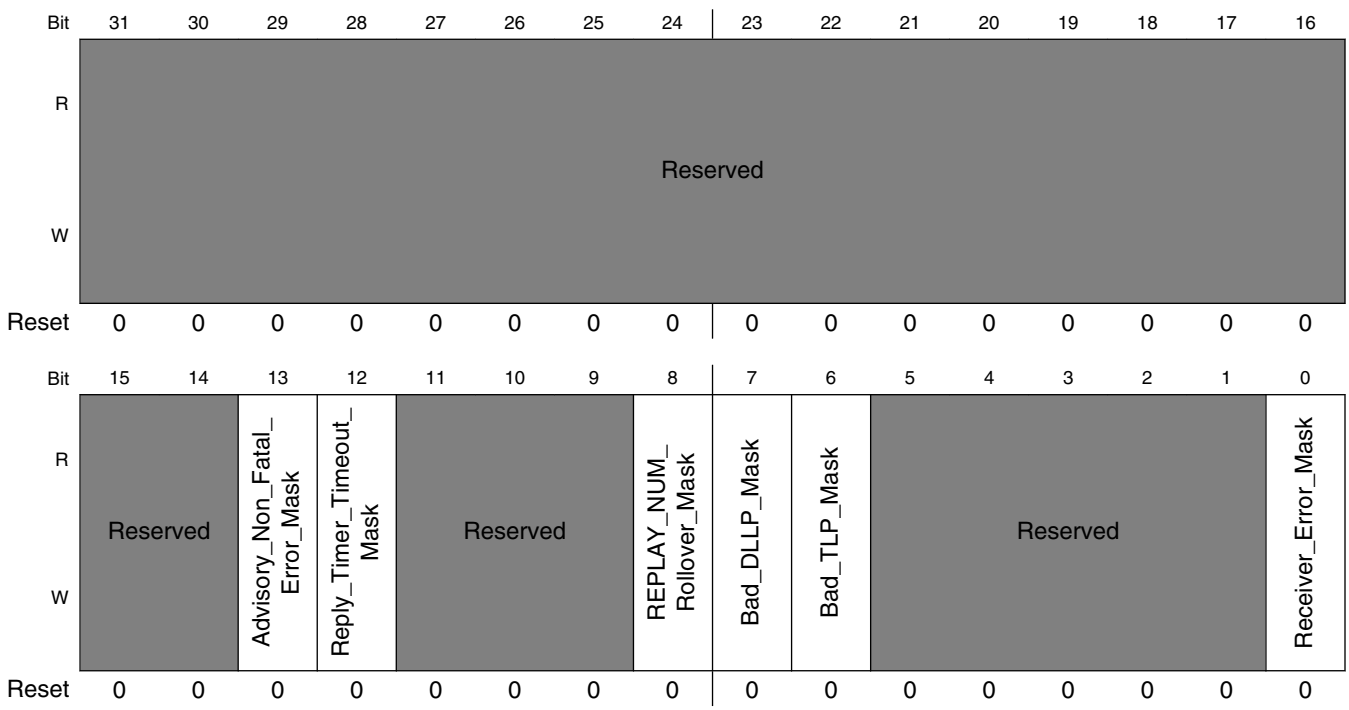
PCIE_EP_CESR field descriptions (continued)

Field	Description
6 Bad_TLP_Status	Bad TLP Status
5–1 -	This field is reserved. Reserved
0 Receiver_Error_Status	Receiver Error Status

48.7.20 Correctable Error Mask Register (PCIE_EP_CEMR)

Offset: 0x14

Address: 1FF_C000h base + 114h offset = 1FF_C114h



PCIE_EP_CEMR field descriptions

Field	Description
31–14 -	This field is reserved. Reserved

Table continues on the next page...

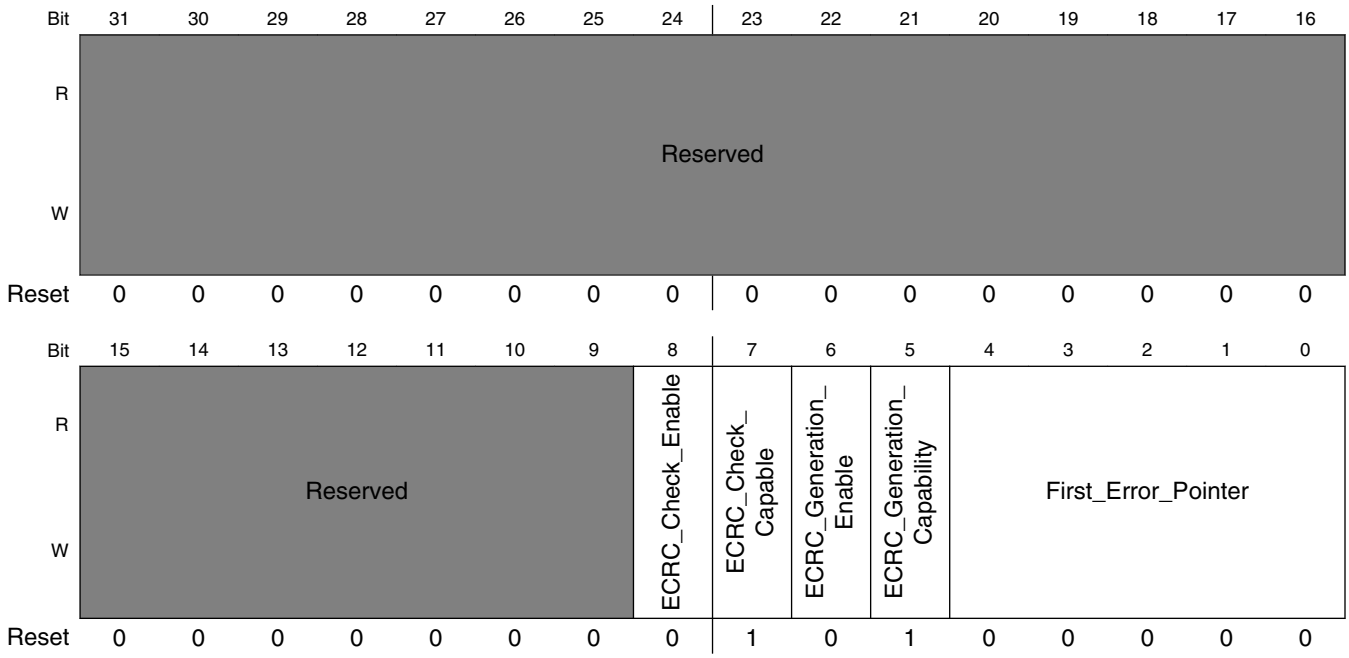
PCIE_EP_CEMR field descriptions (continued)

Field	Description
13 Advisory_Non_Fatal_Error_Mask	Advisory Non-Fatal Error Mask
12 Reply_Timer_Timeout_Mask	Reply Timer Timeout Mask
11–9 -	This field is reserved. Reserved
8 REPLAY_NUM_Rollover_Mask	REPLAY_NUM Rollover Mask
7 Bad_DLLP_Mask	Bad DLLP Mask
6 Bad_TLP_Mask	Bad TLP Mask
5–1 -	This field is reserved. Reserved
0 Receiver_Error_Mask	Receiver Error Mask

48.7.21 Advanced Capabilities and Control Register (PCIE_EP_ACCR)

Offset: 0x18

Address: 1FF_C000h base + 118h offset = 1FF_C118h



PCIE_EP_ACCR field descriptions

Field	Description
31–9 -	This field is reserved. Reserved
8 ECRC_Check_ Enable	ECRC Check Enable
7 ECRC_Check_ Capable	ECRC Check Capable
6 ECRC_ Generation_ Enable	ECRC Generation Enable
5 ECRC_ Generation_ Capability	ECRC Generation Capability
First_Error_ Pointer	First Error Pointer

48.7.22 Header Log Register (PCIE_EP_HLR)

Offset: 0x1C

The Header Log registers collect the header for the TLP corresponding to a detected error. See the PCI Express 3.0 Specification for details. Each of the Header Log registers is type ROS; the default reset value of each Header Log register is 0x00000000.

Address: 1FF_C000h base + 11Ch offset = 1FF_C11Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_EP_HLR field descriptions

Field	Description
-	Header Log Register (nth DWORD)

48.7.23 VC Extended Capability Header (PCIE_EP_VCECHR)

Offset: 0x140

Address: 1FF_C000h base + 140h offset = 1FF_C140h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Next_Capability_Offset												Capability_ Version				Extended_Capability															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0

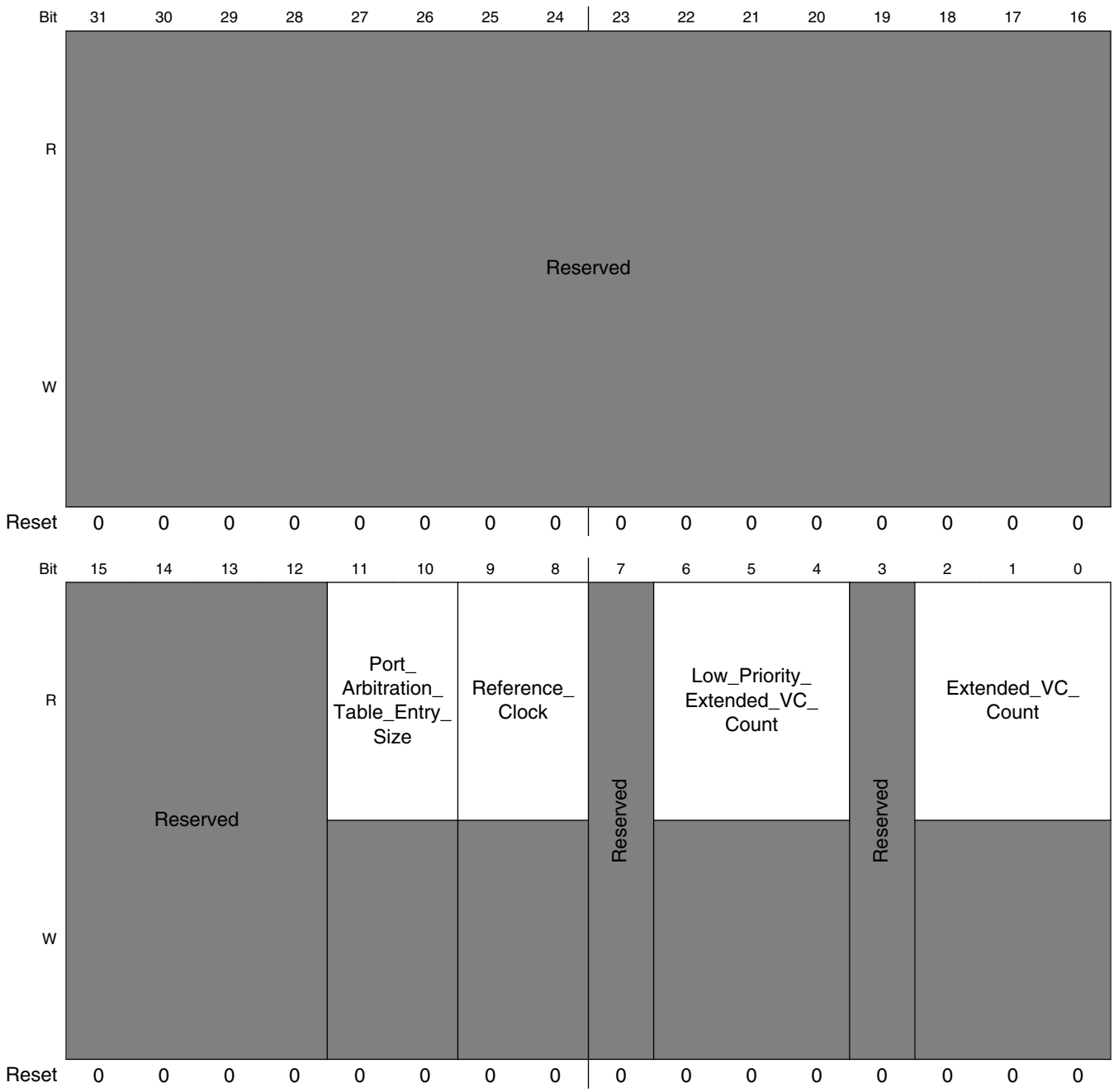
PCIE_EP_VCECHR field descriptions

Field	Description
31–20 Next_Capability_ Offset	Next Capability Offset
19–16 Capability_ Version	Capability Version
Extended_ Capability	PCI Express Extended Capability The default value is 0x2 for VC Capability.

48.7.24 Port VC Capability Register 1 (PCIE_EP_PVCCR1)

Offset: 0x140 + 0x4

Address: 1FF_C000h base + 144h offset = 1FF_C144h



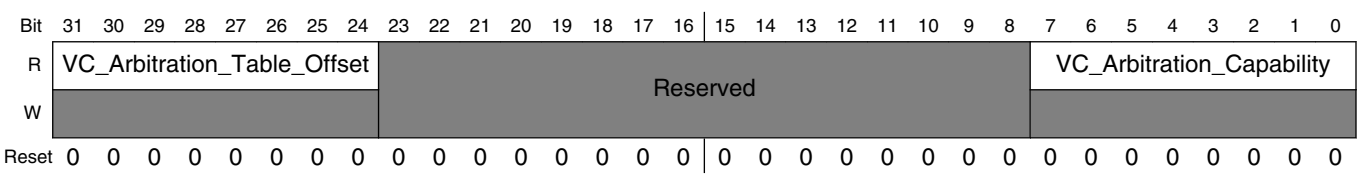
PCIE_EP_PVCCR1 field descriptions

Field	Description
31–12 -	This field is reserved. Reserved
11–10 Port_Arbitration_Table_Entry_Size	Port Arbitration Table Entry Size
9–8 Reference_Clock	Reference Clock
7 -	This field is reserved. Reserved
6–4 Low_Priority_Extended_VC_Count	Low Priority Extended VC Count, writable through the DBI
3 -	This field is reserved. Reserved
Extended_VC_Count	Extended VC Count The default value is the one less than the number of VCs that

48.7.25 Port VC Capability Register 2 (PCIE_EP_PVCCR2)

Offset: 0x140 + 0x8

Address: 1FF_C000h base + 148h offset = 1FF_C148h



PCIE_EP_PVCCR2 field descriptions

Field	Description
31–24 VC_Arbitration_Table_Offset	VC Arbitration Table Offset (not supported) The default value is 0x00 (no arbitration table present).
23–8 -	This field is reserved. Reserved
VC_Arbitration_Capability	VC Arbitration Capability Indicates which VC arbitration mode(s) the device supports, writable through the DBI: •Bit 0: Device supports hardware fixed arbitration scheme. For the core, the scheme is 16-phase weighted round robin (WRR). •Bit 1: Device supports 32-phase WRR

Table continues on the next page...

PCIE_EP_PVCCR2 field descriptions (continued)

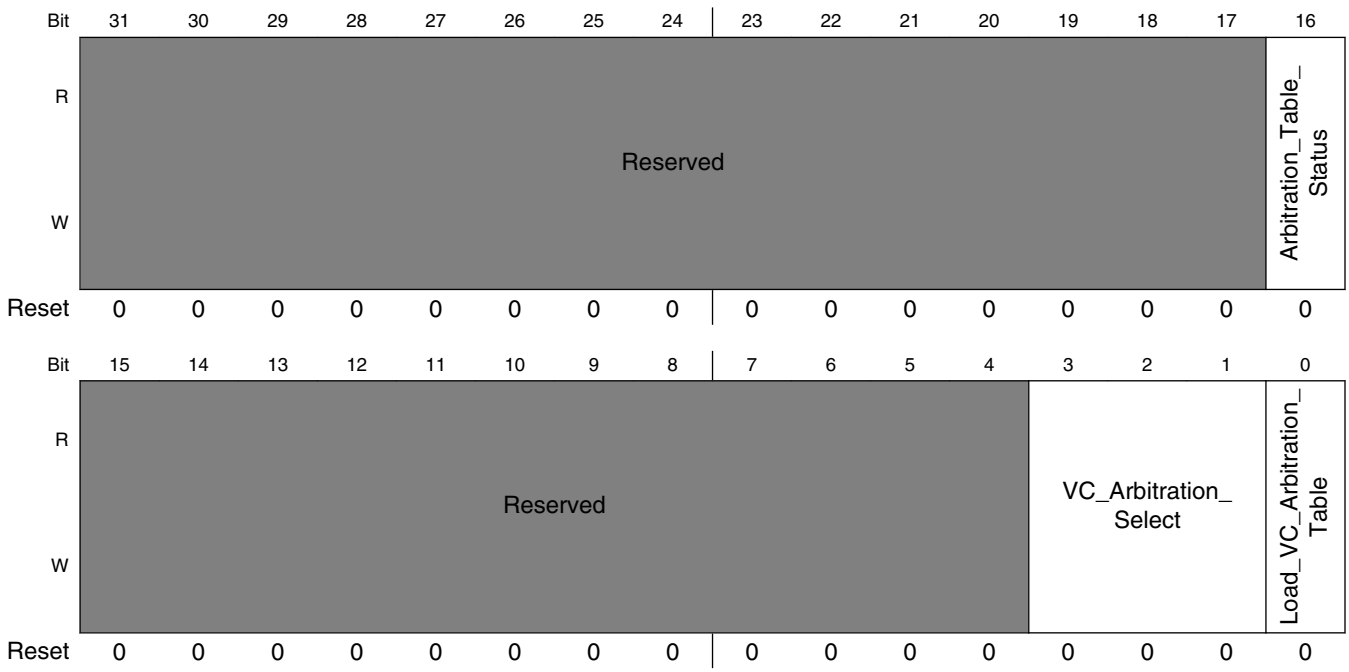
Field	Description
	•Bit 2: Device supports 64-phase WRR •Bit 3: Device supports 128-phase WRR •Bits 4-7: Reserved

48.7.26 Port VC Control and Status Register (PCIE_EP_PVCCSR)

Offset: 0x140 + 0xC

Bytes: 0-1

Address: 1FF_C000h base + 14Ch offset = 1FF_C14Ch



PCIE_EP_PVCCSR field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 Arbitration_Table_Status	Arbitration Table Status
15–4 -	This field is reserved. Reserved
3–1 VC_Arbitration_Select	VC Arbitration Select

Table continues on the next page...

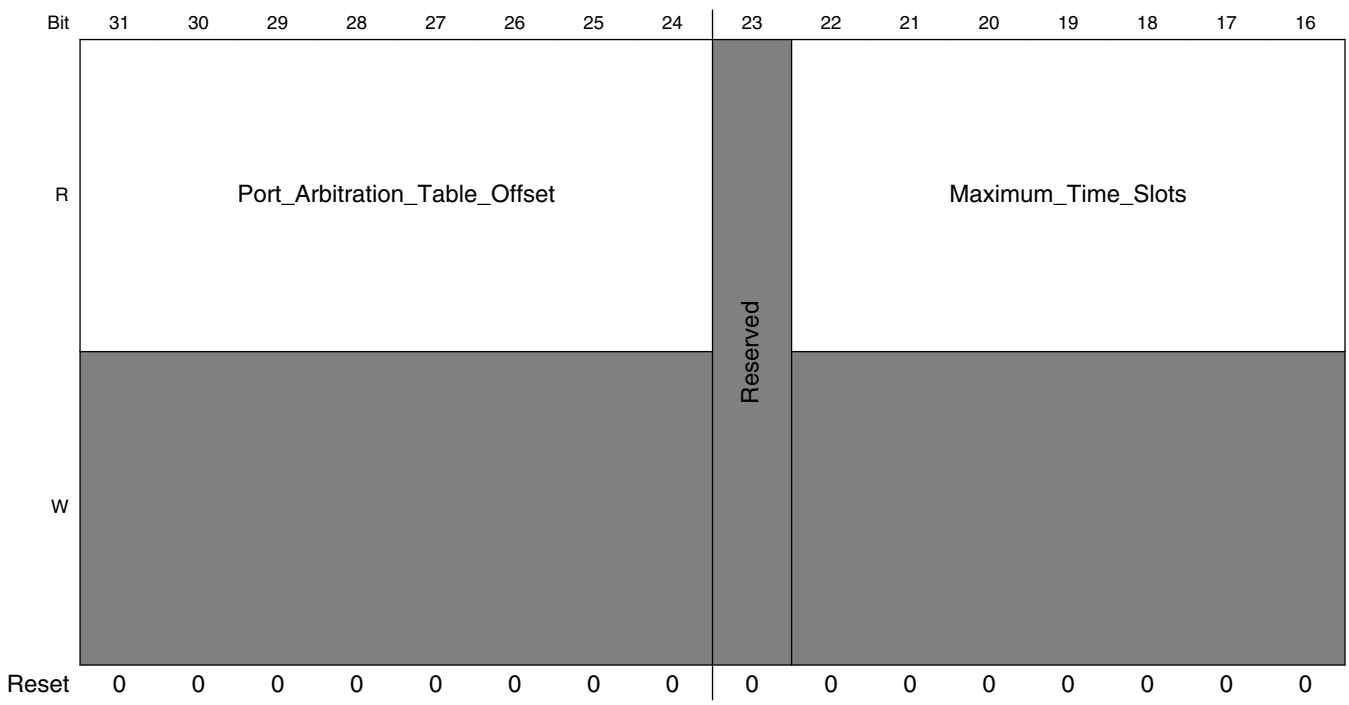
PCIE_EP_PVCCSR field descriptions (continued)

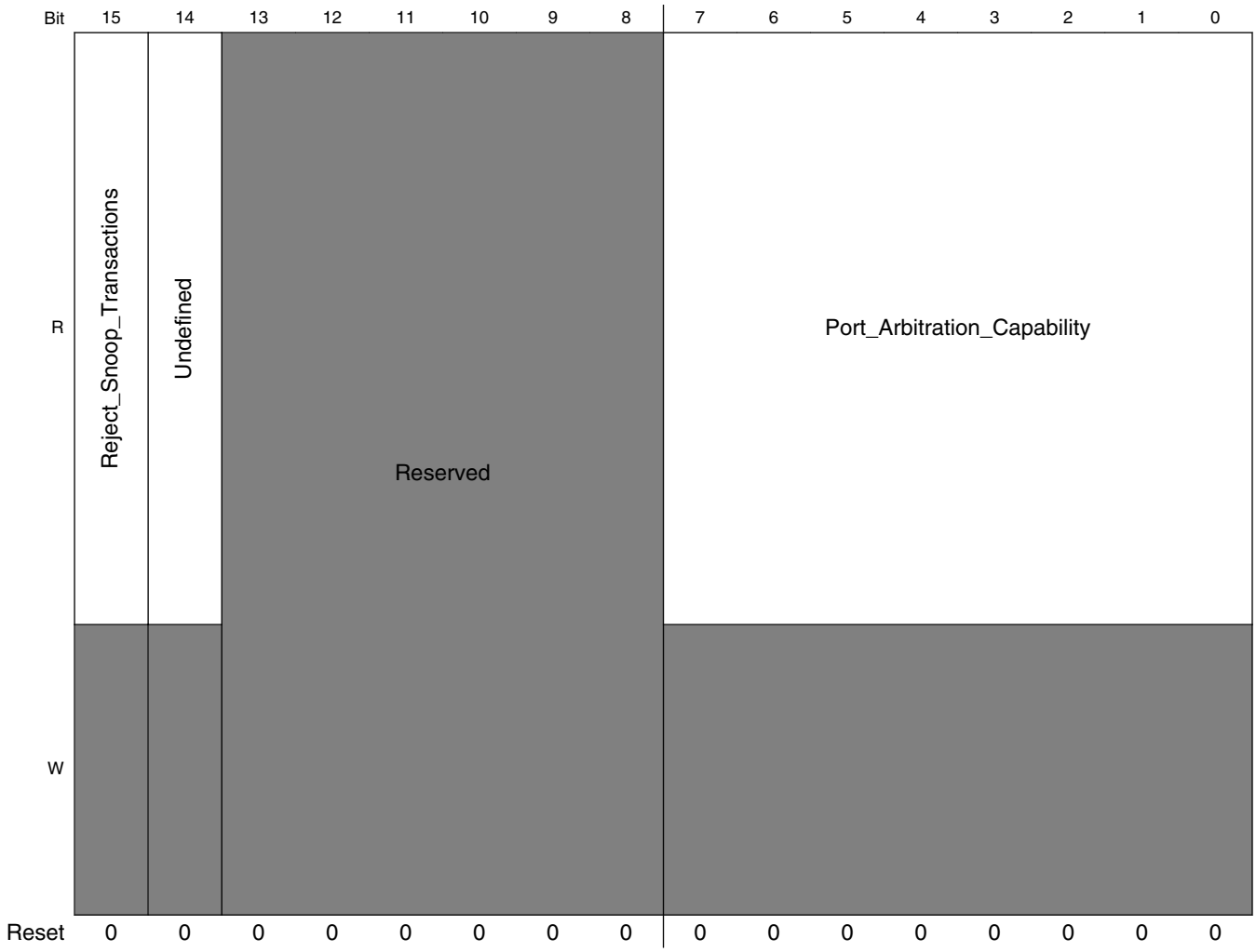
Field	Description
0 Load_VC_ Arbitration_Table	Load VC Arbitration Table

48.7.27 VC Resource Capability Register n (PCIE_EP_VCRCR)

Offset: 0x140 + 0x10

Address: 1FF_C000h base + 150h offset = 1FF_C150h





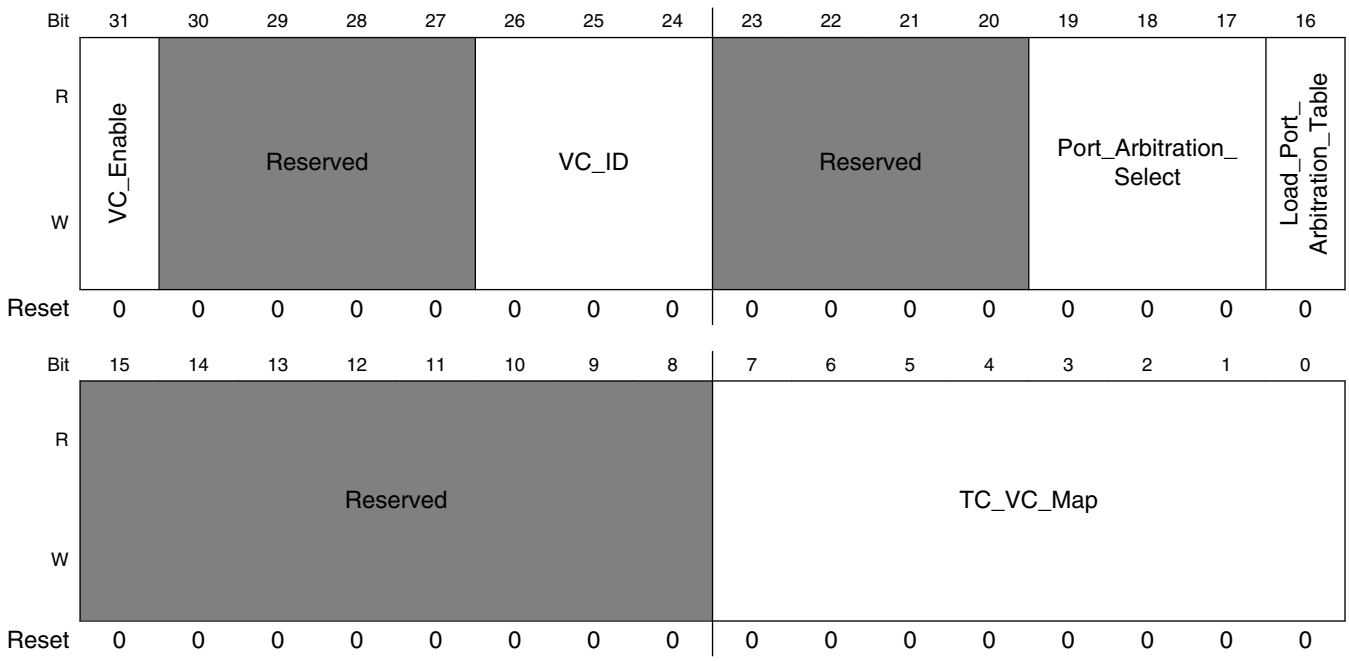
PCIE_EP_VCRCR field descriptions

Field	Description
31–24 Port_Arbitration_Table_Offset	Port Arbitration Table Offset
23 -	This field is reserved. Reserved
22–16 Maximum_Time_Slots	Maximum Time Slots
15 Reject_Snoop_Transactions	Reject Snoop Transactions
14 Undefined	Undefined for PCI Express 1.1 (Was Advanced Packet Switching for PCI Express 1.0a)
13–8 -	This field is reserved. Reserved
Port_Arbitration_Capability	Port Arbitration Capability

48.7.28 VC Resource Control Register n (PCIE_EP_VCRConR)

Offset: 0x140 + 0x14

Address: 1FF_C000h base + 154h offset = 1FF_C154h



PCIE_EP_VCRConR field descriptions

Field	Description
31 VC_Enable	VC Enable Hardwired to 1 for the first VC.
30–27 -	This field is reserved. Reserved
26–24 VC_ID	VC ID Hardwired to 0 for VC0.
23–20 -	This field is reserved. Reserved
19–17 Port_Arbitration_ Select	Port Arbitration Select
16 Load_Port_ Arbitration_Table	Load Port Arbitration Table
15–8 -	This field is reserved. Reserved
TC_VC_Map	TC/VC Map

Table continues on the next page...

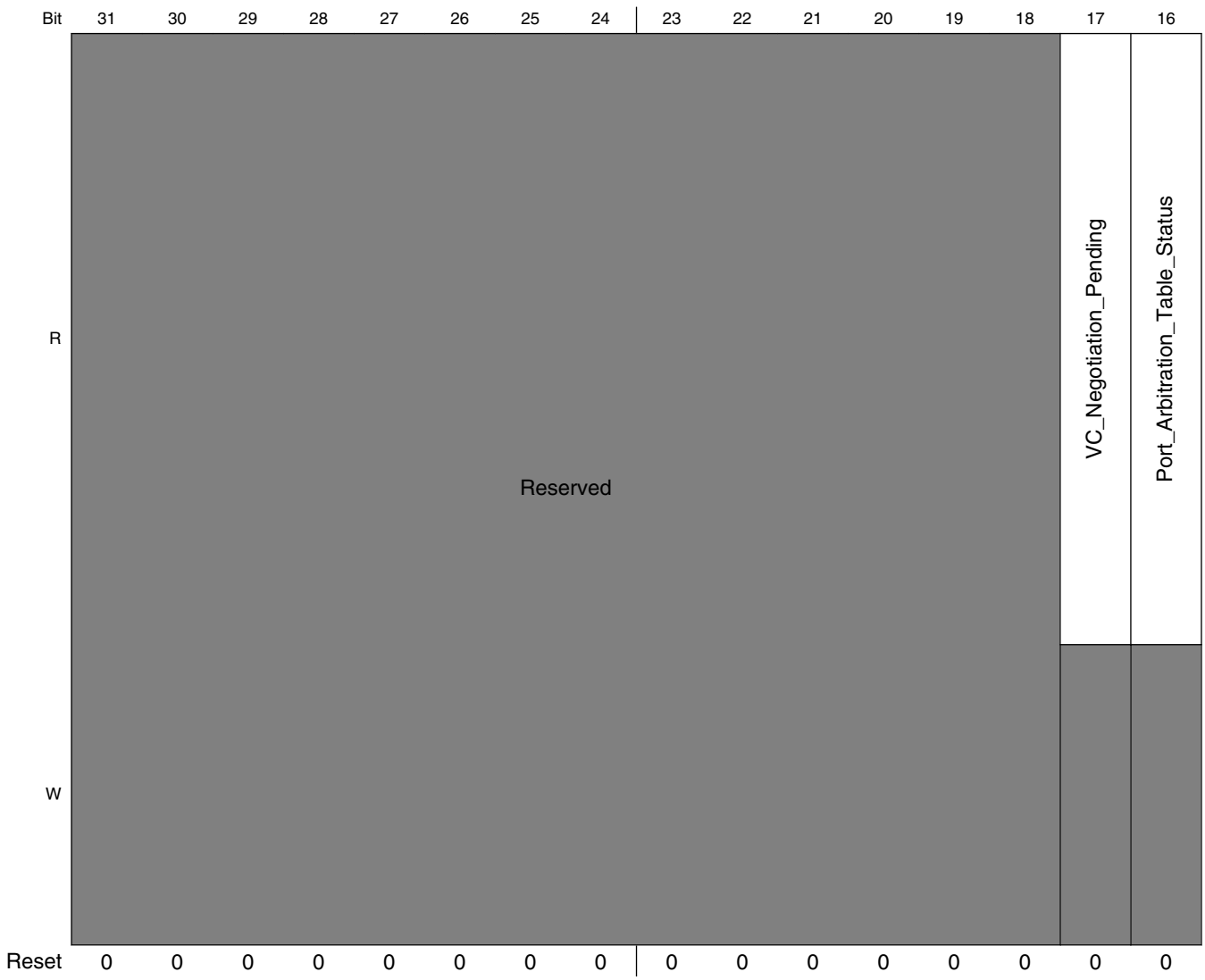
PCIE_EP_VCRConR field descriptions (continued)

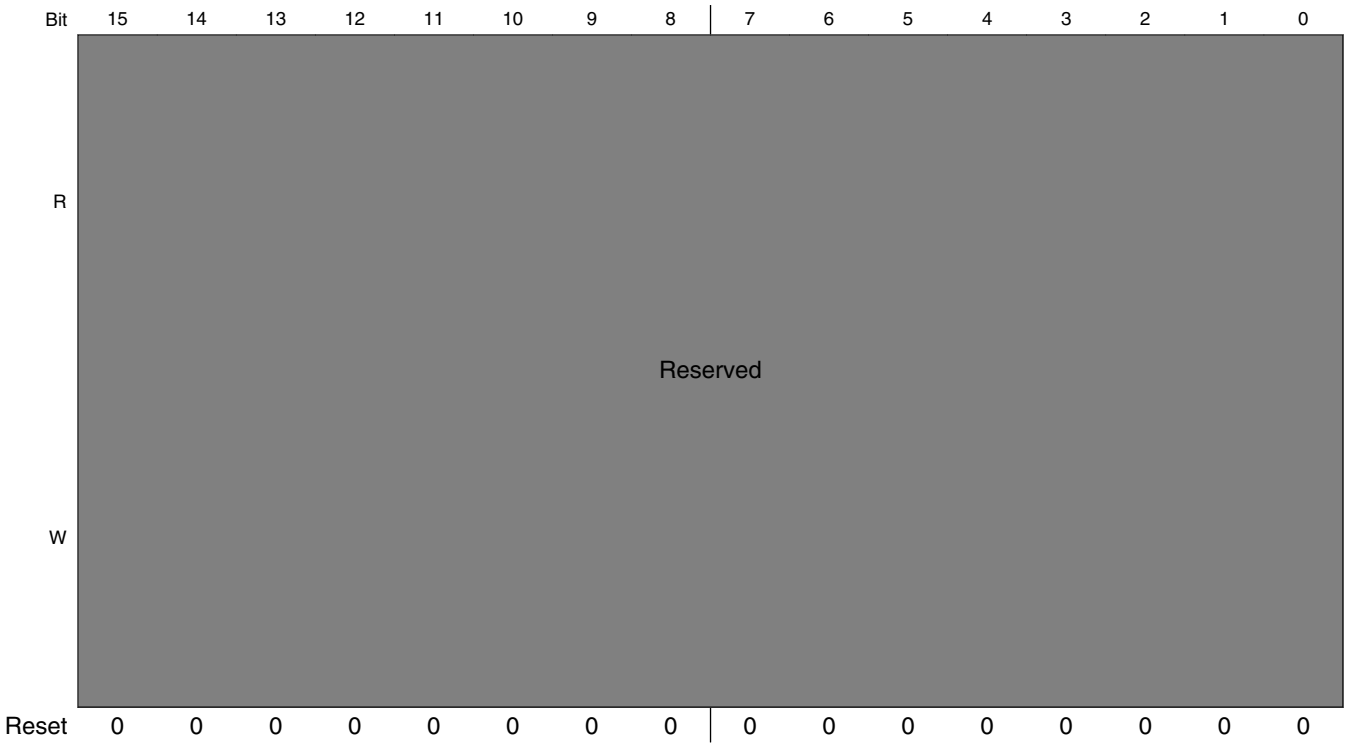
Field	Description
	Bit 0 is hardwired to 1; bits 7:1 are RW.

48.7.29 VC Resource Status Register n (PCIE_EP_VCRSR)

Offset: 0x140 + 0x18

Address: 1FF_C000h base + 158h offset = 1FF_C158h





PCIE_EP_VCRSR field descriptions

Field	Description
31–18 -	This field is reserved. Reserved
17 VC_Negotiation_Pending	VC Negotiation Pending
16 Port_Arbitration_Table_Status	Port Arbitration Table Status
-	This field is reserved. Reserved

48.8 PCIe CTRL RC Mode Memory Map/Register Definition

PCIE_RC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1FF_C000	Device ID and Vendor ID Register (PCIE_RC_DeviceID)	32	R	ABCD _16C3h	48.8.1/4238
1FF_C004	Command and Status Register (PCIE_RC_Command)	32	R/W	0000_0000h	48.8.2/4238

Table continues on the next page...

PCIE_RC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1FF_C008	Revision ID and Class Code Register (PCIE_RC_RevID)	32	R	0000_0000h	48.8.3/4241
1FF_C00C	BIST Register (PCIE_RC_BIST)	32	R/W	0000_0000h	48.8.4/4241
1FF_C010	Base Address 0 (PCIE_RC_BAR0)	32	R	0000_000Ch	48.8.5/4242
1FF_C014	Base Address 1 (PCIE_RC_BAR1)	32	R	0000_0000h	48.8.6/4245
1FF_C018	Bus Number Registers (PCIE_RC_BNR)	32	R	0000_0000h	48.8.7/4245
1FF_C01C	I/O Base Limit Secondary Status Register (PCIE_RC_IOBLSSR)	32	R/W	0000_0000h	48.8.8/4247
1FF_C020	Memory Base and Memory Limit Register (PCIE_RC_MEM_BLR)	32	R/W	0000_0000h	48.8.9/4249
1FF_C024	Prefetchable Memory Base and Limit Register (PCIE_RC_PREF_MEM_BLR)	32	R/W	0000_0000h	48.8.10/ 4250
1FF_C028	Prefetchable Base Upper 32 Bits Register (PCIE_RC_PREF_BASE_U32)	32	R/W	0000_0000h	48.8.11/ 4250
1FF_C02C	Prefetchable Limit Upper 32 Bits Register (PCIE_RC_PREF_LIM_U32)	32	R/W	0000_0000h	48.8.12/ 4251
1FF_C030	I/O Base and Limit Upper 16 Bits Register (PCIE_RC_IO_BASE_LIM_U16)	32	R/W	0000_0000h	48.8.13/ 4251
1FF_C034	Capability Pointer Register (PCIE_RC_CAPPR)	32	R	0000_0040h	48.8.14/ 4252
1FF_C038	Expansion ROM Base Address Register (PCIE_RC_EROMBAR)	32	R/W	0000_0000h	48.8.15/ 4252
1FF_C038	Expansion ROM BAR Mask Register (PCIE_RC_EROMMASK)	32	R/W	0000_0000h	48.8.16/ 4253
1FF_C040	Power Management Capability Register (PCIE_RC_PMCR)	32	R	DBC3_5001h	48.8.17/ 4254
1FF_C044	Power Management Control and Status Register (PCIE_RC_PMCSR)	32	R/W	0000_0004h	48.8.18/ 4257
1FF_C070	PCI Express Capability ID Register (PCIE_RC_CIDR)	32	R	0000_0000h	48.8.19/ 4258
1FF_C074	Device Capabilities Register (PCIE_RC_DCR)	32	R/W	0000_0000h	48.8.20/ 4261
1FF_C078	Device Control Register (PCIE_RC_DConR)	32	R/W	0000_0000h	48.8.21/ 4263
1FF_C07C	Link Capabilities Register (PCIE_RC_LCR)	32	R	0000_0000h	48.8.22/ 4265
1FF_C080	Link Control and Status Register (PCIE_RC_LCSR)	32	R/W	0000_0000h	48.8.23/ 4268
1FF_C084	Slot Capabilities Register (PCIE_RC_SCR)	32	R	0000_0000h	48.8.24/ 4270
1FF_C088	Slot Control and Status Register (PCIE_RC_SCSR)	32	R/W	0000_0000h	48.8.25/ 4273
1FF_C08C	Root Control and Capabilities Register (PCIE_RC_RCCR)	32	R/W	0000_0000h	48.8.26/ 4276

Table continues on the next page...

PCIE_RC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1FF_C090	Root Status Register (PCIE_RC_RSR)	32	w1c	0000_0000h	48.8.27/ 4278
1FF_C094	Device Capabilities 2 Register (PCIE_RC_DCR2)	32	R	0000_001Fh	48.8.28/ 4279
1FF_C098	Device Control and Status 2 Register (PCIE_RC_DCSR2)	32	R/W	0000_0000h	48.8.29/ 4281
1FF_C09C	Link Capabilities 2 Register (PCIE_RC_LCR2)	32	R	0000_0000h	48.8.30/ 4283
1FF_C0A0	Link Control and Status 2 Register (PCIE_RC_LCSR2)	32	R/W	0000_0000h	48.8.31/ 4285
1FF_C100	AER Capability Header (PCIE_RC_AER)	32	R/W	0000_0000h	48.8.32/ 4287
1FF_C104	Uncorrectable Error Status Register (PCIE_RC_UESR)	32	R/W	0000_0000h	48.8.33/ 4288
1FF_C108	Uncorrectable Error Mask Register (PCIE_RC_UEMR)	32	R/W	0000_0000h	48.8.34/ 4290
1FF_C10C	Uncorrectable Error Severity Register (PCIE_RC_UESevR)	32	R/W	000C_2031h	48.8.35/ 4292
1FF_C110	Correctable Error Status Register (PCIE_RC_CESR)	32	R/W	0000_0000h	48.8.36/ 4294
1FF_C114	Correctable Error Mask Register (PCIE_RC_CEMR)	32	R/W	0000_0000h	48.8.37/ 4295
1FF_C118	Advanced Capabilities and Control Register (PCIE_RC_ACCR)	32	R/W	0000_00A0h	48.8.38/ 4297
1FF_C11C	Header Log Register (PCIE_RC_HLR)	32	R	0000_0000h	48.8.39/ 4298
1FF_C12C	Root Error Command Register (PCIE_RC_RECR)	32	R/W	0000_0000h	48.8.40/ 4299
1FF_C130	Root Error Status Register (PCIE_RC_RESR)	32	R/W	0000_0000h	48.8.41/ 4300
1FF_C134	Error Source Identification Register (PCIE_RC_ESIR)	32	R	0000_0000h	48.8.42/ 4301
1FF_C140	VC Extended Capability Header (PCIE_RC_VCECHR)	32	R	0000_0012h	48.8.43/ 4302
1FF_C144	Port VC Capability Register 1 (PCIE_RC_PVCCR1)	32	R	0000_0000h	48.8.44/ 4303
1FF_C148	Port VC Capability Register 2 (PCIE_RC_PVCCR2)	32	R	0000_0000h	48.8.45/ 4304
1FF_C14C	Port VC Control and Status Register (PCIE_RC_PVCCSR)	32	R/W	0000_0000h	48.8.46/ 4305
1FF_C150	VC Resource Capability Register n (PCIE_RC_VCRCR)	32	R	0000_0000h	48.8.47/ 4307
1FF_C154	VC Resource Control Register n (PCIE_RC_VCRConR)	32	R/W	0000_0000h	48.8.48/ 4309

Table continues on the next page...

PCIE_RC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1FF_C158	VC Resource Status Register n (PCIE_RC_VCRSR)	32	R	0000_0000h	48.8.49/ 4311

48.8.1 Device ID and Vendor ID Register (PCIE_RC_DeviceID)

Offset: 0x00

The default values of both Device ID and Vendor ID are hardware configuration parameters. The application can overwrite the default values of both Device ID and Vendor ID through the DBI.

Address: 1FF_C000h base + 0h offset = 1FF_C000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																Reserved															
Reset	1	0	1	0	1	0	1	1	1	1	0	0	1	1	0	1	0	0	0	1	0	1	1	0	1	1	0	0	0	0	1	1

PCIE_RC_DeviceID field descriptions

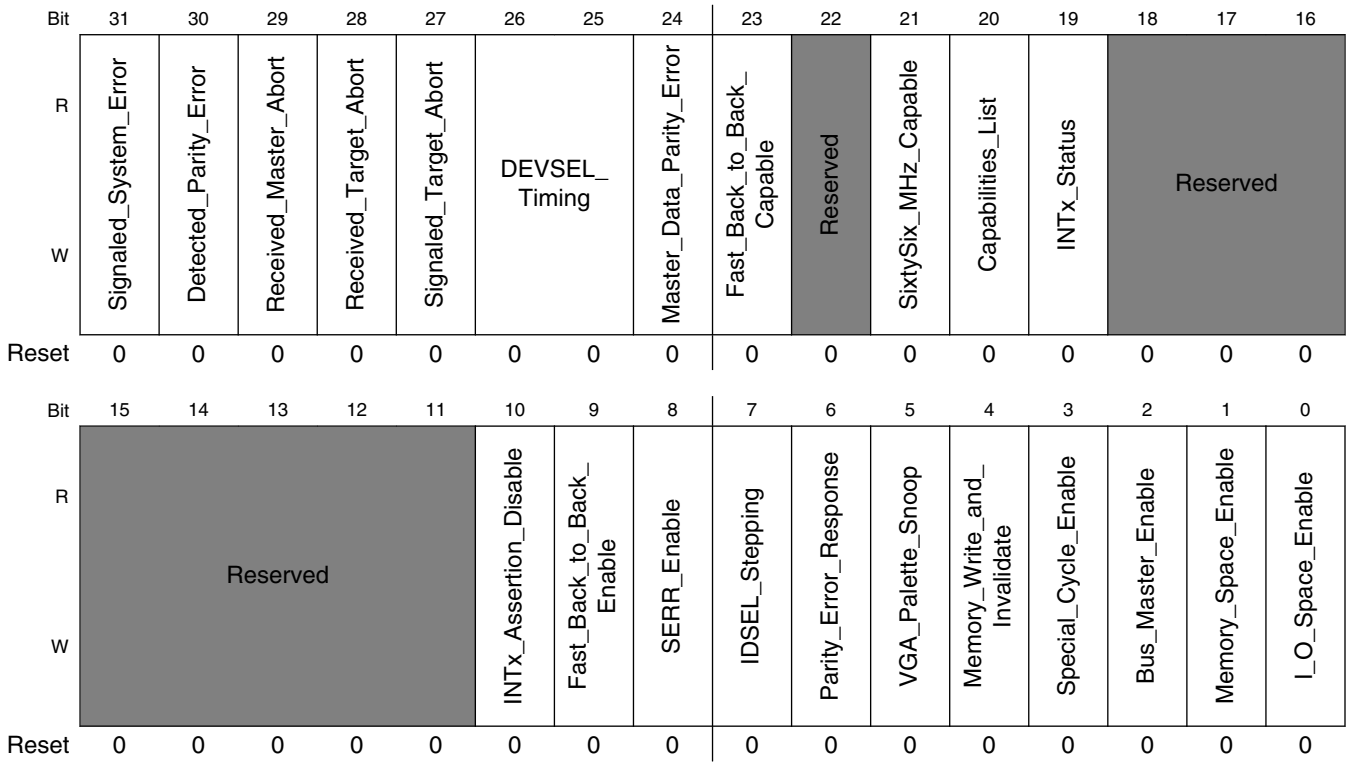
Field	Description
31–16 -	This field is reserved. Reserved
-	This field is reserved. Reserved

48.8.2 Command and Status Register (PCIE_RC_Command)

Offset: 0x04

Bytes: 0-1

Address: 1FF_C000h base + 4h offset = 1FF_C004h



PCIE_RC_Command field descriptions

Field	Description
31 Signaled_ System_Error	Signaled System Error
30 Detected_Parity_ Error	Detected Parity Error
29 Received_ Master_Abort	Received Master Abort
28 Received_ Target_Abort	Received Target Abort
27 Signaled_Target_ Abort	Signaled Target Abort
26–25 DEVSEL_Timing	DEVSEL Timing Not applicable for PCI Express. Hardwired to 0.
24 Master_Data_ Parity_Error	Master Data Parity Error
23 Fast_Back_to_ Back_Capable	Fast Back-to-Back Capable Not applicable for PCI Express. Hardwired to 0.

Table continues on the next page...

PCIE_RC_Command field descriptions (continued)

Field	Description
22 -	This field is reserved. Reserved
21 SixtySix_MHz_Capable	66 MHz Capable Not applicable for PCI Express. Hardwired to 0.
20 Capabilities_List	Capabilities List Indicates presence of an extended capability item. Hardwired to 1.
19 INTx_Status	INTx Status
18–16 -	This field is reserved. Reserved
15–11 -	This field is reserved. Reserved
10 INTx_Assertion_Disable	INTx Assertion Disable
9 Fast_Back_to_Back_Enable	Fast Back-to-Back Enable Not applicable for PCI Express. Must be hardwired to 0.
8 SERR_Enable	SERR# Enable
7 IDSEL_Stepping	IDSEL Stepping/Wait Cycle Control Not applicable for PCI Express. Must be hardwired to 0
6 Parity_Error_Response	Parity Error Response
5 VGA_Palette_Snoop	VGA Palette Snoop Not applicable for PCI Express. Must be hardwired to 0.
4 Memory_Write_and_Invalidate	Memory Write and Invalidate Not applicable for PCI Express. Must be hardwired to 0.
3 Special_Cycle_Enable	Special Cycle Enable Not applicable for PCI Express. Must be hardwired to 0.
2 Bus_Master_Enable	Bus Master Enable
1 Memory_Space_Enable	Memory Space Enable
0 I_O_Space_Enable	I/O Space Enable

48.8.3 Revision ID and Class Code Register (PCIE_RC_RevID)

Offset: 0x08

Byte: 0

Address: 1FF_C000h base + 8h offset = 1FF_C008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BASE_CLASS_CODE_N								SUB_CLASS_CODE_N								IF_CODE_N								CX_REVISION_ID_N							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_RC_RevID field descriptions

Field	Description
31–24 BASE_CLASS_CODE_N	Base Class Code, writable through the DBI
23–16 SUB_CLASS_CODE_N	Subclass Code, writable through the DBI
15–8 IF_CODE_N	Programming Interface, writable through the DBI
CX_REVISION_ID_N	Revision ID, writable through the DBI

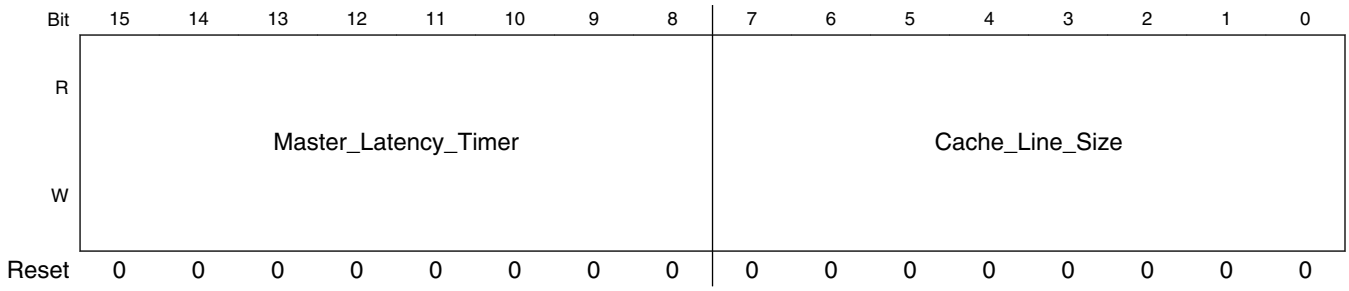
48.8.4 BIST Register (PCIE_RC_BIST)

Offset: 0x0C

Byte: 0

Address: 1FF_C000h base + Ch offset = 1FF_C00Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Not_supported_by__core								Multi_Function_Device	Configuration_Header_Format						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



PCIE_RC_BIST field descriptions

Field	Description
31–24 Not_supported_ by__core	The BIST register functions are not supported by the core. All 8 bits of the BIST register are hardwired to 0.
23 Multi_Function_ Device	Multi Function Device The default value is 0 for a single function device (`CX_NFUNC = 1) or 1 for a multi-function device (`CX_NFUNC != 1). The Multi Function Device bit is writable through the DBI.
22–16 Configuration_ Header_Format	Configuration Header Format Hardwired to 0 for type 0.
15–8 Master_Latency_ Timer	Master Latency Timer Not applicable for PCI Express, hardwired to 0.
Cache_Line_Size	Cache Line Size The Cache Line Size register is RW for legacy compatibility purposes and is not applicable to PCI Express device functionality. Writing to the Cache Line Size register does not impact functionality of the core.

48.8.5 Base Address 0 (PCIE_RC_BAR0)

Offset: 0x10-0x24

The core provides three pairs of 32-bit BARs for each implemented function. Each pair (BARs 0 and 1, BARs 2 and 3, BARs 4 and 5) can be configured as follows:

- One 64-bit BAR: For example, BARs 0 and 1 are combined to form a single 64-bit BAR.
- Two 32-bit BARs: For example, BARs 0 and 1 are two independent 32-bit BARs.
- One 32-bit BAR: For example, BAR 0 is a 32-bit BAR and BAR 1 is either disabled or removed from the core altogether to reduce gate count.

In addition, you can configure each BAR to have its incoming Requests routed to either:

- RTRGT1
-

The following sections describe how to set up the BAR types and sizes by programming values into the base address registers. For more information about routing Requests to either RTRGT1 on a BAR-by- BAR basis, see Receive Filtering.

The contents of the six BARs determine the BAR configuration. The reset values of the BARs are determined by hardware configuration options.

At runtime, application software can overwrite the BAR contents to reconfigure the BARs (unless the affected BAR is removed during hardware configuration). Application software must observe the rules listed below when writing to the BARs.

The rules for BAR configuration are the same for all three pairs. Using BARs 0 and 1 as the example pair, the rules for BAR configuration are:

- Any pair (for example, BARs 0 and 1) can be configured as one 64-bit BAR, two 32-bit BARs, or one 32-bit BAR.
- BAR pairs cannot overlap to form a 64-bit BAR. For example, you cannot combine BARs 1 and 2 to form a 64-bit BAR.
-
- An I/O BAR must be a 32-bit BAR and cannot be prefetchable.
- If the device is configured as a PCI Express Endpoint (not a Legacy Endpoint), then any memory that is configured as prefetchable must be a 64-bit memory BAR.
- If BAR 0 is configured as a 64-bit BAR:

- BAR 1 is the upper 32 bits of the combined 64-bit BAR formed by BARs 0 and 1. Therefore, BAR 1 must be disabled and cannot be configured independently.
- BAR 0 must be a memory BAR and can be either prefetchable or non-prefetchable.
- The contents of the BAR 0 Mask register determine the number of writable bits in the 64-bit BAR, subject to the restrictions described in BAR Mask Register. The BAR 1 Mask register contains the upper 32 bits of the BAR 0 Mask value.
- BAR 0 can be disabled by writing 0 to bit 0 of the BAR 0 Mask register

- If BAR 0 is configured as a 32-bit BAR:

- You can configure BAR 1 as an independent 32-bit BAR
- BAR 0 can be configured as a memory BAR or an I/O BAR.
- The contents of the BAR 0 Mask register determine the number of writable bits in the 32-bit BAR 0, subject to the restrictions described in BAR Mask Registers.
- BAR 0 can be disabled by writing 0 to bit 0 of the BAR 0 Mask register

- When BAR 0 is configured as a 32-bit BAR, BAR 1 is available as an independent 32-bit BAR according to the following rules:

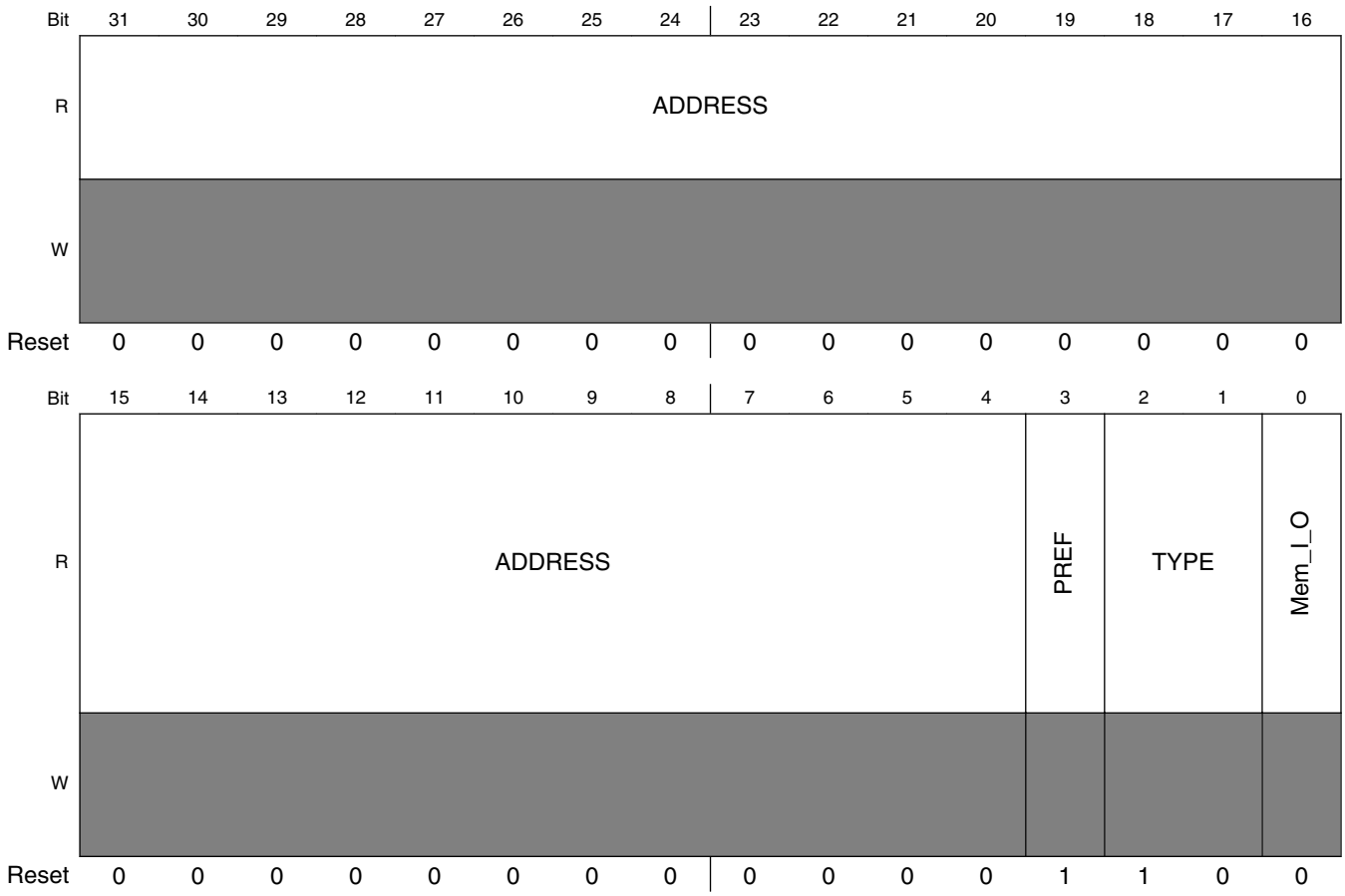
- BAR 1 can be configured as a memory BAR or an I/O BAR.

- The contents of the BAR 1 Mask register determine the number of writable bits in the 32-bit BAR 1, subject to the restrictions described in BAR Mask Registers.
-
-

The same rules apply for pairs 2/3 and 4/5.

Offset: 0x10 (if included in the core hardware configuration)

Address: 1FF_C000h base + 10h offset = 1FF_C010h



PCIE_RC_BAR0 field descriptions

Field	Description
31–4 ADDRESS	BAR 0 base address bits (for a 64-bit BAR, the remaining upper address bits are in BAR 1). The BAR 0 Mask value determines which address bits are masked.
3 PREF	If BAR 0 is an I/O BAR, bit 3 is the second least significant bit of the base address. Bits [3:0] are writable through the DBI. If BAR 0 is a memory BAR, bit 3 indicates if the memory region is prefetchable: 0 = Non-prefetchable 1 = Prefetchable

Table continues on the next page...

PCIE_RC_BAR0 field descriptions (continued)

Field	Description
2–1 TYPE	If BAR 0 is an I/O BAR, bit 2 the least significant bit of the base address and bit 1 is 0. Bits [3:0] are writable through the DBI. If BAR 0 is a memory BAR, bits [2:1] determine the BAR type: 00 = 32-bit BAR 10 = 64-bit BAR
0 Mem_I_O	Bits [3:0] are writable through the DBI. 0 = BAR 0 is a memory BAR 1 = BAR 0 is an I/O BAR

48.8.6 Base Address 1 (PCIE_RC_BAR1)

Address: 0x14

Address: 1FF_C000h base + 14h offset = 1FF_C014h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_RC_BAR1 field descriptions

Field	Description
ADDRESS	BAR 1 contains the upper 32 bits of the BAR 0 base address (bits [63:32]).

48.8.7 Bus Number Registers (PCIE_RC_BNR)

Address: 1FF_C000h base + 18h offset = 1FF_C018h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SECONDARY_LAT_TMR								SUBORD_BUS_NUM								SECONDARY_BUS_NUM								PRIMARY_BUS_NUM							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

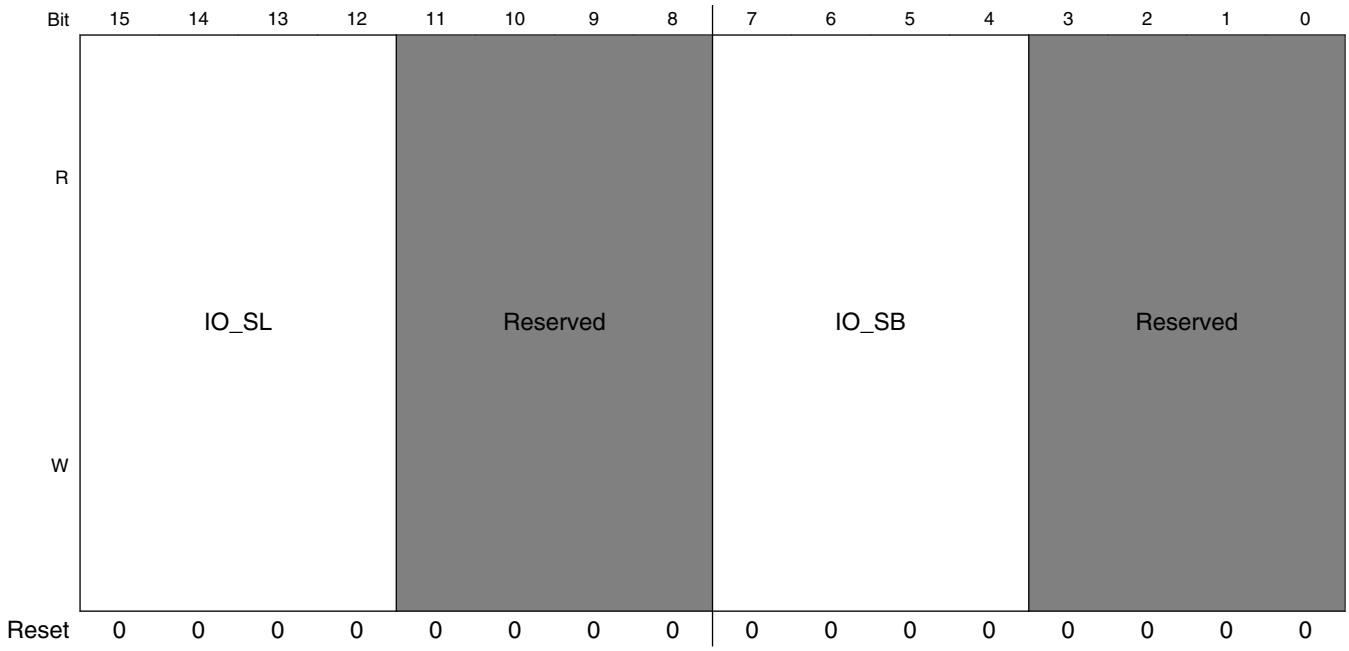
PCIE_RC_BNR field descriptions

Field	Description
31–24 SECONDARY_ LAT_TMR	Secondary latency timer.
23–16 SUBORD_BUS_ NUM	Subordinate bus number.
15–8 SECONDARY_ BUS_NUM	Secondary bus number.
PRIMARY_BUS_ NUM	Primary bus number.

48.8.8 I/O Base Limit Secondary Status Register (PCIE_RC_IOBLSSR)

Address: 1FF_C000h base + 1Ch offset = 1FF_C01Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DET_PARITY_ERR	RX_SYS_ERR	RX_MASTER_ABORT	RX_TARGET_ABORT	SIG_TARGET_ABORT	Reserved		MSTR_DAT_PARITY_ERR	FAST_B2B_CAP	Reserved	66M_CAP	Reserved				
W	w1c	w1c	w1c	w1c	w1c			w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



PCIE_RC_IOBLSSR field descriptions

Field	Description
31 DET_PARITY_ERR	Detected Parity Error.
30 RX_SYS_ERR	Received System Error.
29 RX_MASTER_ABORT	Received Master Abort.
28 RX_TARGET_ABORT	Received Target Abort.
27 SIG_TARGET_ABORT	Signaled Target Abort.
26–25 -	This field is reserved. Reserved.
24 MSTR_DAT_PARITY_ERR	Master Data Parity Error.
23 FAST_B2B_CAP	Reserved.
22 -	This field is reserved. Reserved.
21 66M_CAP	66 MHz Capable. Not applicable to PCI Express, hardwired to 0.
20–16 -	This field is reserved. Reserved.

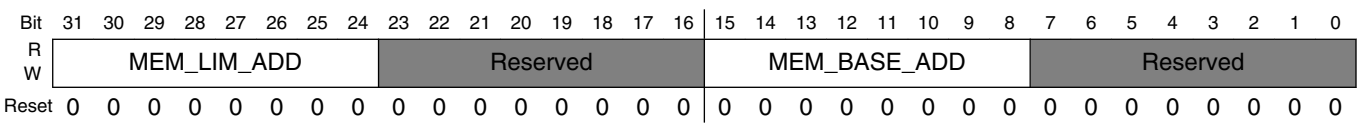
Table continues on the next page...

PCIE_RC_IOBLSSR field descriptions (continued)

Field	Description
15–12 IO_SL	I/O Space Limit.
11–8 -	This field is reserved. Reserved.
7–4 IO_SB	I/O Space Base.
-	This field is reserved. Reserved.

48.8.9 Memory Base and Memory Limit Register (PCIE_RC_MEM_BLR)

Address: 1FF_C000h base + 20h offset = 1FF_C020h



PCIE_RC_MEM_BLR field descriptions

Field	Description
31–24 MEM_LIM_ADD	Memory Limit Address.
23–16 -	This field is reserved. Reserved.
15–8 MEM_BASE_ ADD	Memory Base Address.
-	This field is reserved. Reserved.

48.8.10 Prefetchable Memory Base and Limit Register (PCIE_RC_PREF_MEM_BLR)

Address: 1FF_C000h base + 24h offset = 1FF_C024h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	UPPER12_END_ADD												Reserved				UPPER12_START_ADD												Reserved			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_RC_PREF_MEM_BLR field descriptions

Field	Description
31–20 UPPER12_END_ADD	Upper 12 bits of 32-bit Prefetchable Memory End Address.
19–16 -	This field is reserved. Reserved.
15–4 UPPER12_START_ADD	Upper 12 bits of 32-bit Prefetchable Memory Start Address.
-	This field is reserved. Reserved.

48.8.11 Prefetchable Base Upper 32 Bits Register (PCIE_RC_PREF_BASE_U32)

Prefetchable Base Upper 32 Bits Register

Address: 1FF_C000h base + 28h offset = 1FF_C028h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	UPPER32_BASE_PREF_MEM_ADD																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_RC_PREF_BASE_U32 field descriptions

Field	Description
UPPER32_BASE_PREF_MEM_ADD	Upper 32 Bits of Base Address of Prefetchable Memory Space. Used only when 64-bit prefetchable memory addressing is enabled.

48.8.12 Prefetchable Limit Upper 32 Bits Register (PCIE_RC_PREF_LIM_U32)

Prefetchable Limit Upper 32 Bits Register

Address: 1FF_C000h base + 2Ch offset = 1FF_C02Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	UPPER32_LIM_PREF_MEM_ADD																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_RC_PREF_LIM_U32 field descriptions

Field	Description
UPPER32_LIM_PREF_MEM_ADD	Upper 32 Bits of Limit Address of Prefetchable Memory Space. Used only when 64-bit prefetchable memory addressing is enabled.

48.8.13 I/O Base and Limit Upper 16 Bits Register (PCIE_RC_IO_BASE_LIM_U16)

I/O Base and Limit Upper 16 Bits Register

Address: 1FF_C000h base + 30h offset = 1FF_C030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	UPPER16_IO_LIM																UPPER16_IO_BASE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_RC_IO_BASE_LIM_U16 field descriptions

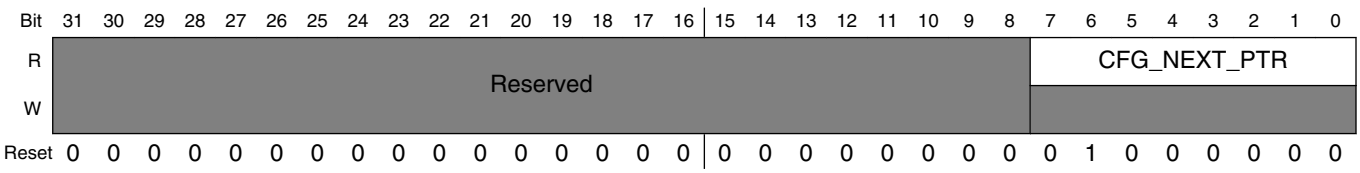
Field	Description
31–16 UPPER16_IO_LIM	Upper 16 Bits of I/O Limit (if 32-bit I/O decoding is supported for devices on the secondary side).
UPPER16_IO_BASE	Upper 16 Bits of I/O Base (if 32-bit I/O decoding is supported for devices on the secondary side).

48.8.14 Capability Pointer Register (PCIE_RC_CAPPR)

Offset: 0x34

Byte: 0

Address: 1FF_C000h base + 34h offset = 1FF_C034h



PCIE_RC_CAPPR field descriptions

Field	Description
31–8 -	This field is reserved. Reserved
CFG_NEXT_PTR	First Capability Pointer.

48.8.15 Expansion ROM Base Address Register (PCIE_RC_EROMBAR)

Offset: 0x38

Address: 1FF_C000h base + 38h offset = 1FF_C038h

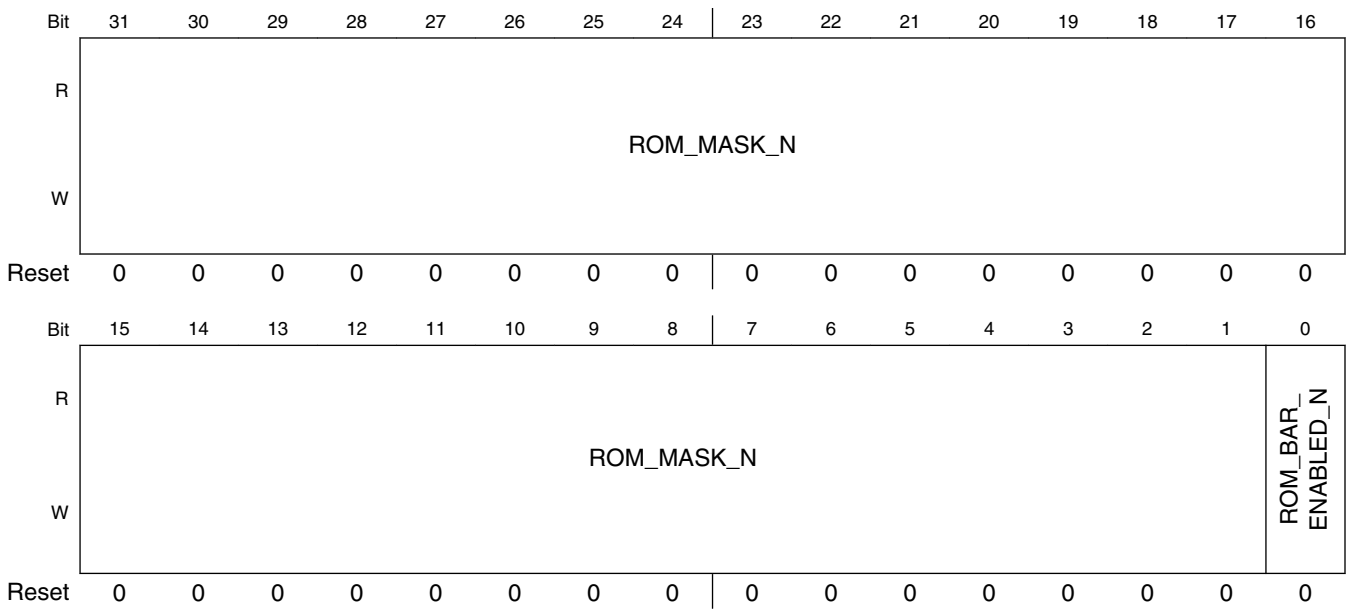
PCIE_RC_EROMBAR field descriptions

Field	Description
31–11 ADDRESS	Expansion ROM Address
10–1 -	This field is reserved. Reserved
0 ENABLE	Expansion ROM Enable

48.8.16 Expansion ROM BAR Mask Register (PCIE_RC_EROMMASK)

Offset: 0x38 (same as the Expansion ROM BAR, but requires dbi_cs2 for write access)

Address: 1FF_C000h base + 38h offset = 1FF_C038h



PCIE_RC_EROMMASK field descriptions

Field	Description
31–1 ROM_MASK_N	Indicates which Expansion ROM BAR bits to mask (make non-writable) from host software, which, in turn, determines the size of the BAR. For example, writing 0xFFF to the Expansion ROM BAR Mask register claims a 4096-byte BAR by masking bits 11:0 of the BAR from writing by host software. The maximum value is 0xFFFFFF because the maximum space that can be claimed by an Expansion ROM BAR is 16 MB. The Expansion ROM BAR Mask register is invisible to host software and not readable from the application. Application access depends on the value of

Table continues on the next page...

PCIE_RC_EROMMASK field descriptions (continued)

Field	Description
	• the Expansion ROM BAR Mask register is writable through the DBI. •
0 ROM_BAR_ ENABLED_N	Expansion ROM BAR Enable 0 Expansion ROM BAR is disabled 1 Expansion ROM BAR is enabled

48.8.17 Power Management Capability Register (PCIE_RC_PMCR)

The core implements power management capabilities. The Capability Pointer field in the configuration header points to the PCI Power Management registers as the first extended capability by default.

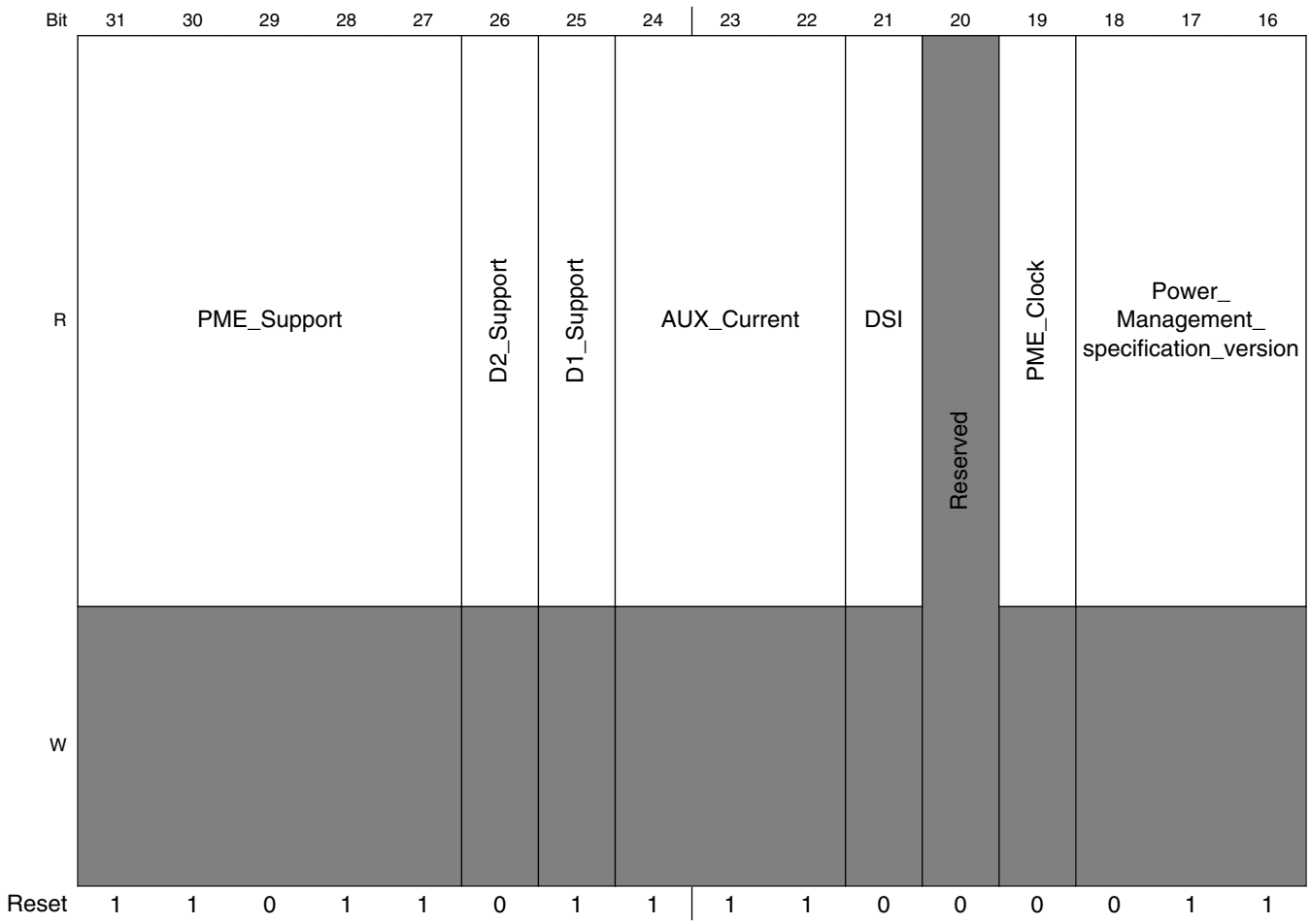
The extent of the power management implementation in the core includes:

- Power Management register space
- Link state information (provided to both the application logic and PHY interfaces)
- Power management-ready clock and reset implementation

The following sections describe the PCI Power Management registers implemented in the core. See the *PCI Power Management specification* and the *PCI Express 3.0 Specification* for more details.

Offset: `CFG_PM_CAP

Address: 1FF_C000h base + 40h offset = 1FF_C040h



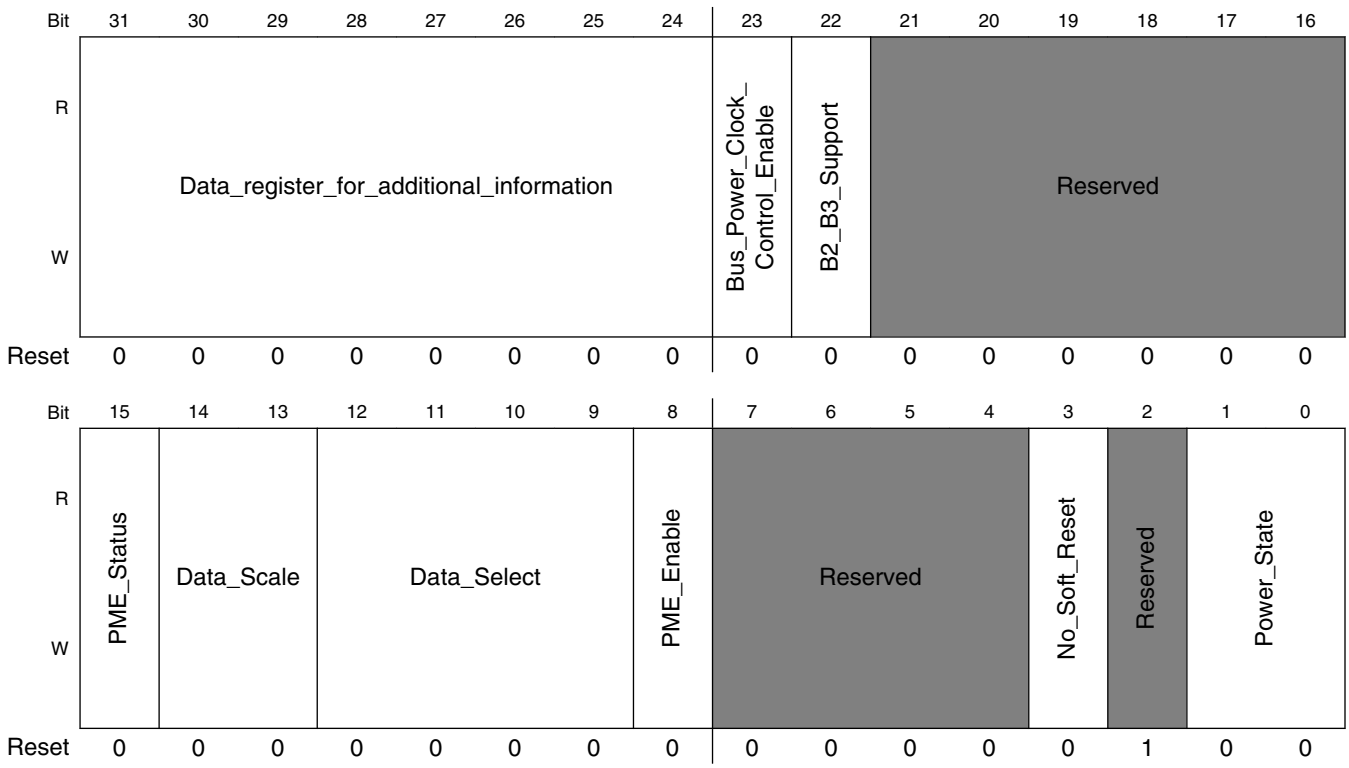
PCIE_RC_PMCR field descriptions

Field	Description
31–27 PME_Support	PME_Support Identifies the power states from which the core can generate PME Messages. A value of 0 for any bit indicates that the device (or function) is not capable of generating PME Messages while in that power state: • Bit 11: If set, PME Messages can be generated from D0 • Bit 12: If set, PME Messages can be generated from D1 • Bit 13: If set, PME Messages can be generated from D2 • Bit 14: If set, PME Messages can be generated from $D3_{hot}$ • Bit 15: If set, PME Messages can be generated from $D3_{cold}$ The PME_Support field is writable through the DBI.
26 D2_Support	D2 Support, writable through the DBI
25 D1_Support	D1 Support, writable through the DBI
24–22 AUX_Current	AUX Current, writable through the DBI
21 DSI	Device Specific Initialization (DSI), writable through the DBI
20 -	This field is reserved. Reserved
19 PME_Clock	PME Clock, hardwired to 0
18–16 Power_ Management_ specification_ version	Power Management specification version, writable through the DBI
15–8 Next_Capability_ Pointer	Next Capability Pointer See and .
Power_ Management_ Capability_ID	Power Management Capability ID

48.8.18 Power Management Control and Status Register (PCIE_RC_PMCSR)

Offset: `CFG_PM_CAP + 0x04

Address: 1FF_C000h base + 44h offset = 1FF_C044h



PCIE_RC_PMCSR field descriptions

Field	Description
31–24 Data_register_for_additional_information	Data register for additional information (not supported)
23 Bus_Power_Clock_Control_Enable	Bus Power/Clock Control Enable, hardwired to 0
22 B2_B3_Support	B2/B3 Support, hardwired to 0
21–16 -	This field is reserved. Reserved
15 PME_Status	PME Status

Table continues on the next page...

PCIE_RC_PMCSR field descriptions (continued)

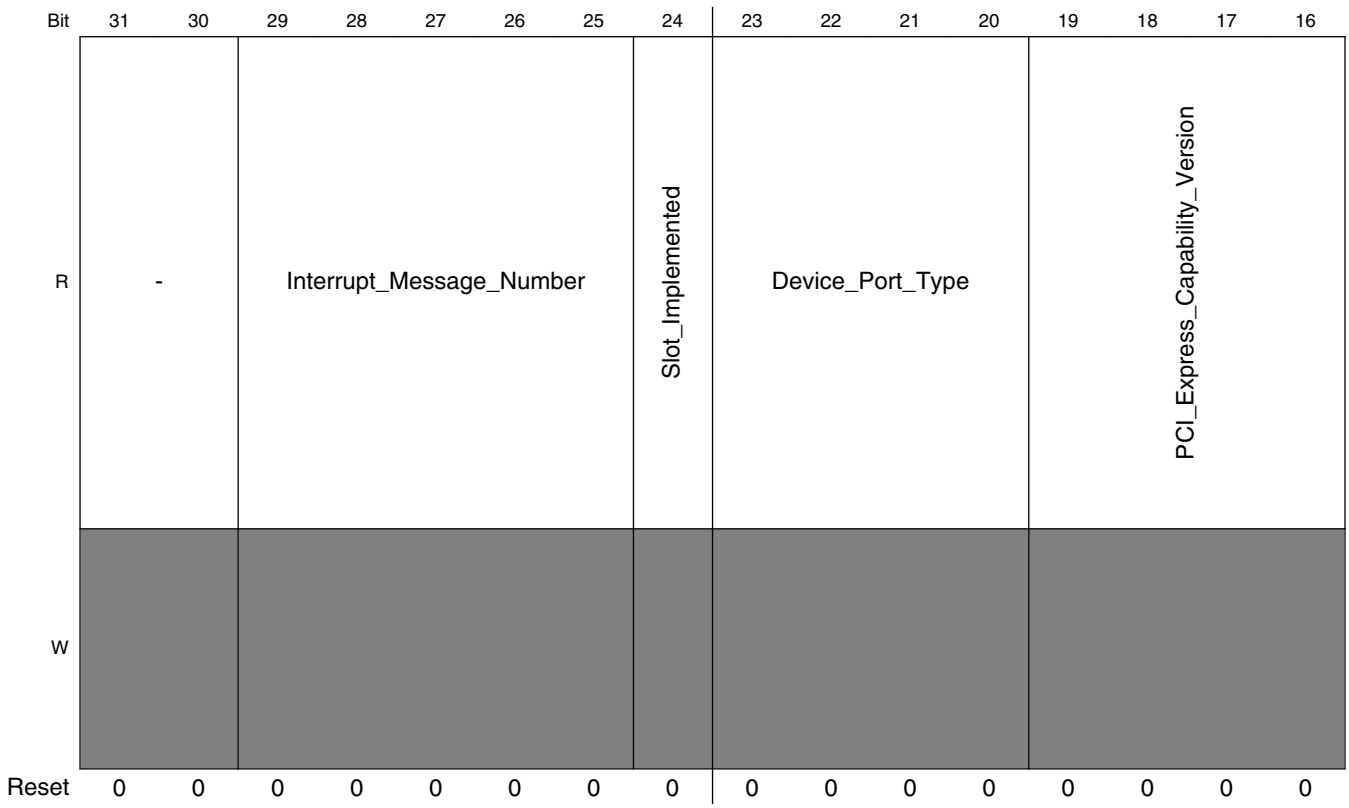
Field	Description
	Indicates if a previously enabled PME event occurred or not.
14–13 Data_Scale	Data Scale (not supported)
12–9 Data_Select	Data Select (not supported)
8 PME_Enable	PME Enable (sticky bit) A value of 1 indicates that the device is enabled to generate PME.
7–4 -	This field is reserved. Reserved
3 No_Soft_Reset	No Soft Reset, writable through the DBI
2 -	This field is reserved. Reserved
Power_State	Power State The written value is ignored if the specific state is not supported. Controls the device power state: 00 D0 01 D1 10 D2 11 D3

48.8.19 PCI Express Capability ID Register (PCIE_RC_CIDR)

The core implements the PCI Express Capability Structure as defined in the PCI Express 3.0 Specification.

Offset: CFG_PCIE_CAP + 0x00

Address: 1FF_C000h base + 70h offset = 1FF_C070h



PCIE_RC_CIDR field descriptions

Field	Description
31–30 -	RsvdP

Table continues on the next page...

PCIE_RC_CIDR field descriptions (continued)

Field	Description
29–25 Interrupt_ Message_ Number	Interrupt Message Number Updated by hardware, writable through the DBI.
24 Slot_ Implemented	Slot Implemented, writable through the DBI
23–20 Device_Port_ Type	Device/Port Type Indicates the specific type of this PCI Express Function. Supported encodings for RC and DM(RC mode) are: •4'b0100: Root Port of PCI Express Root Complex **NOTE:** Note: All other encodings (including those for PCI/PCI-X bridges and RC Integrated Endpoint) are NOT supported.
19–16 PCI_Express_ Capability_ Version	PCI Express Capability Version
15–8 Next_Capability_ Pointer	Next Capability Pointer
PCI_Express_ Capability_ID	PCI Express Capability ID

48.8.20 Device Capabilities Register (PCIE_RC_DCR)

Offset: `CFG_PCIE_CAP + 0x04

Address: 1FF_C000h base + 74h offset = 1FF_C074h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved				Captured_ Slot_Power_ Limit_Scale		Captured_Slot_Power_Limit_Value								Reserved	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Role_Based_Error_ Reporting	Reserved	Reserved	Reserved	Endpoint_L1_ Acceptable_Latency			Endpoint_L0s_ Acceptable_Latency			Extended_Tag_Field_ Supported	Phantom_ Function_ Supported		Max_Payload_Size_ Supported		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_RC_DCR field descriptions

Field	Description
31–28 -	This field is reserved. Reserved
27–26 Captured_Slot_ Power_Limit_ Scale	Captured Slot Power Limit Scale Upstream port only.
25–18 Captured_Slot_ Power_Limit_ Value	Captured Slot Power Limit Value Upstream port only.
17–16 -	This field is reserved. Reserved
15 Role_Based_ Error_Reporting	Role-Based Error Reporting, writable through the DBI. Required to be set for device compliant to 1.1 spec and later.

Table continues on the next page...

PCIE_RC_DCR field descriptions (continued)

Field	Description
14 -	This field is reserved. Reserved Undefined since PCI Express 1.1 (Was Power Indicator Present for PCI Express 1.0a)
13 -	This field is reserved. Reserved Undefined since PCI Express 1.1 (Was Attention Indicator Present for PCI Express 1.0a)
12 -	This field is reserved. Reserved Undefined since PCI Express 1.1 (Was Attention Button Present for PCI Express 1.0a)
11–9 Endpoint_L1_Acceptable_Latency	Endpoint L1 Acceptable Latency Must be 0x0 for non-Endpoint devices.
8–6 Endpoint_L0s_Acceptable_Latency	Endpoint L0s Acceptable Latency Must be 0x0 for non-Endpoint devices.
5 Extended_Tag_Field_Supported	Extended Tag Field Supported This bit is writable through the DBI. However, if the core supports only 5 bits of TAG, then the application must not write a 1 to this field because the hardware to support more than 32 tags are not implemented.
4–3 Phantom_Function_Supported	Phantom Function Supported This field is writable through the DBI. However, Phantom Function is not supported. Therefore, the application must not write any value other than 0x0 to this field.
Max_Payload_Size_Supported	Max_Payload_Size Supported, writable through the DBI

48.8.21 Device Control Register (PCIE_RC_DConR)

Offset: `CFG_PCIE_CAP + 0x08

Address: 1FF_C000h base + 78h offset = 1FF_C078h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved										Transaction_Pending	Aux_Power_Detected	Unsupported_Request_Detected	Fatal_Error_Detected	Non_Fatal_Error_detected	Correctable_Error_Detected
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved	Max_Read_Request_Size			Enable_No_Snoop	AUX_Power_PM_Enable	Phantom_Function_Enable	Extended_Tag_Field_Enable	Max_Payload_Size			Enable_Relaxed_Ordering	Unsupported_Request_Reporting_Enable	Fatal_Error_Reporting_Enable	Non_Fatal_Error_Reporting_Enable	Correctable_Error_Reporting_Enable
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_RC_DConR field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21 Transaction_Pending	Transaction Pending Hard-wired to 0.
20 Aux_Power_Detected	Aux Power Detected From sys_aux_pwr_det input port.
19 Unsupported_Request_Detected	Unsupported Request Detected Errors are logged in this register regardless of whether error reporting is enabled in the Device Control register.
18 Fatal_Error_Detected	Fatal Error Detected Errors are logged in this register regardless of whether error reporting is enabled in the Device Control register.

Table continues on the next page...

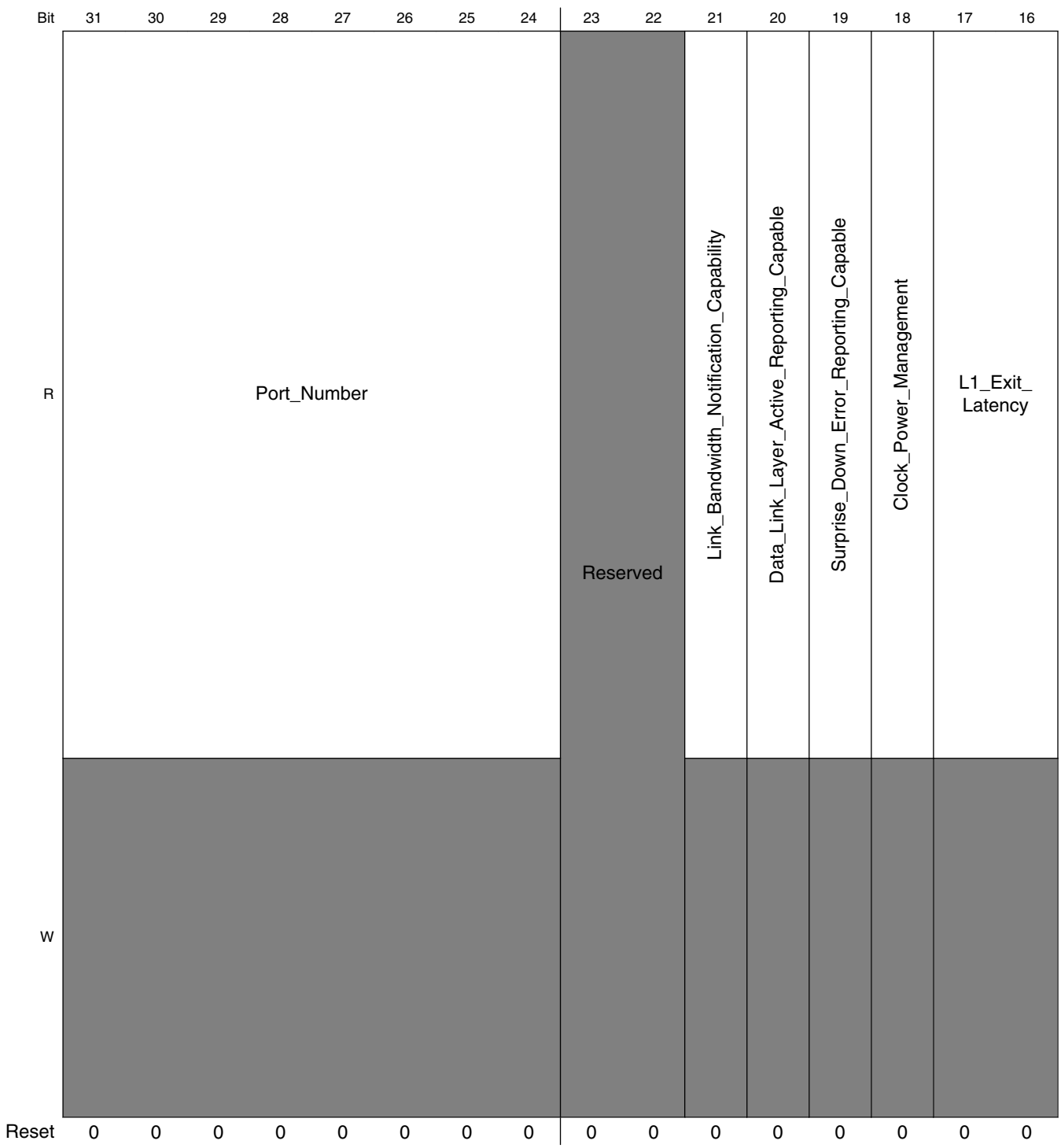
PCIE_RC_DConR field descriptions (continued)

Field	Description
17 Non_Fatal_ Error_detected	Non-Fatal Error detected Errors are logged in this register regardless of whether error reporting is enabled in the Device Control register.
16 Correctable_ Error_Detected	Correctable Error Detected Errors are logged in this register regardless of whether error reporting is enabled in the Device Control register.
15 -	This field is reserved. Reserved
14–12 Max_Read_ Request_Size	Max_Read_Request_Size
11 Enable_No_ Snoop	Enable No Snoop
10 AUX_Power_ PM_Enable	AUX Power PM Enable
9 Phantom_ Function_Enable	Phantom Function Enable
8 Extended_Tag_ Field_Enable	Extended Tag Field Enable
7–5 Max_Payload_ Size	Max_Payload_Size
4 Enable_Relaxed_ Ordering	Enable Relaxed Ordering
3 Unsupported_ Request_ Reporting_ Enable	Unsupported Request Reporting Enable
2 Fatal_Error_ Reporting_ Enable	Fatal Error Reporting Enable
1 Non_Fatal_ Error_Reporting_ Enable	Non-Fatal Error Reporting Enable
0 Correctable_ Error_Reporting_ Enable	Correctable Error Reporting Enable

48.8.22 Link Capabilities Register (PCIE_RC_LCR)

Offset: `CFG_PCIE_CAP + 0x0C

Address: 1FF_C000h base + 7Ch offset = 1FF_C07Ch



Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	L1_Exit_Latency	L0s_Exit_Latency			Active_State_Link_PM_Support		Maximum_Link_Width						Max_Link_Speeds			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_RC_LCR field descriptions

Field	Description
31–24 Port_Number	Port Number
23–22 -	This field is reserved. Reserved
21 Link_Bandwidth_Notification_Capability	Link Bandwidth Notification Capability Hardwired to 1 for Downstream Ports and 0 for Upstream Ports.
20 Data_Link_Layer_Active_Reporting_Capable	Data Link Layer Active Reporting Capable Hardwired to 1 for Downstream Ports and 0 for Upstream Ports.
19 Surprise_Down_Error_Reporting_Capable	Surprise Down Error Reporting Capable Not supported, hardwired to 0x0.

Table continues on the next page...

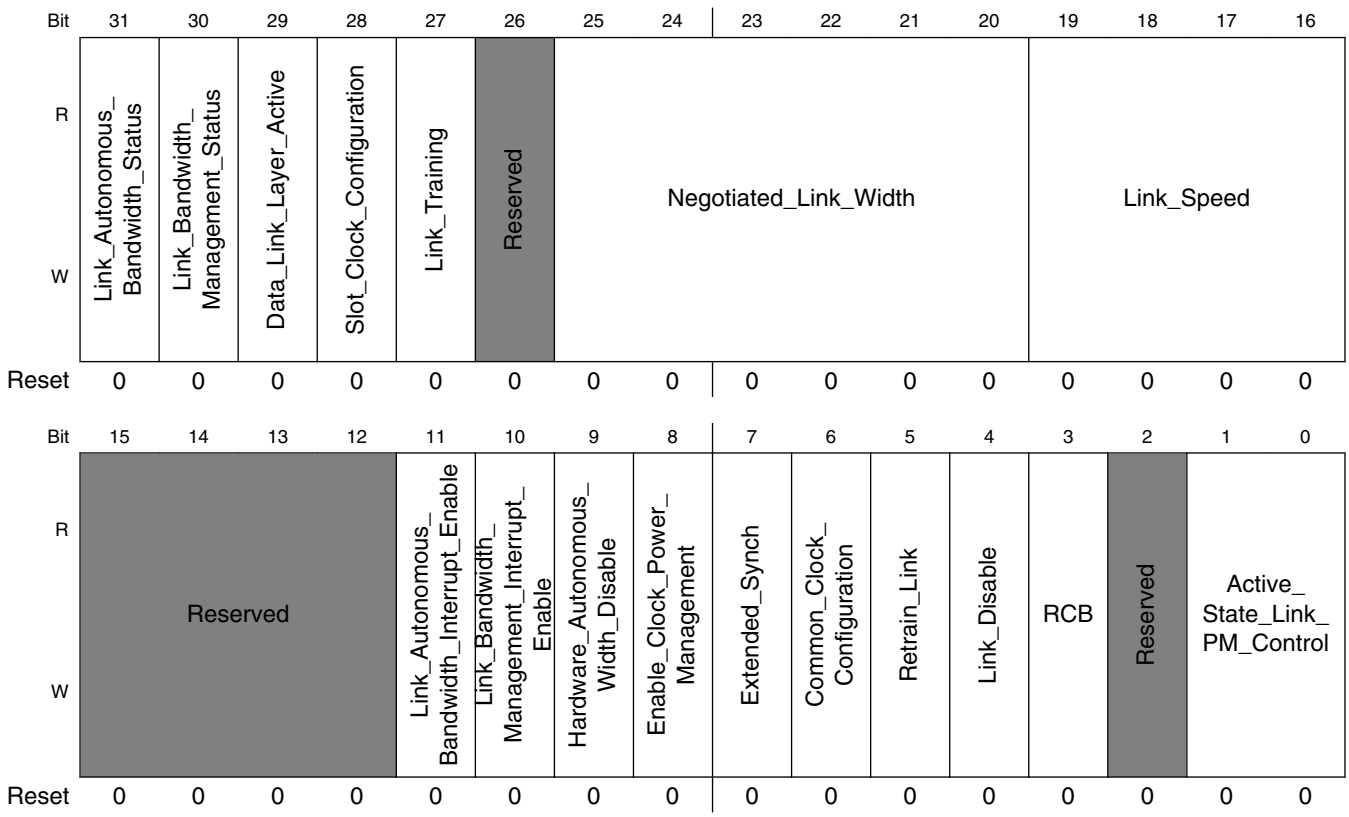
PCIE_RC_LCR field descriptions (continued)

Field	Description
18 Clock_Power_Management	Clock Power Management Component can tolerate the removal of refclk via CLKREQ# (if supported). Hardwired to 0 for downstream ports. Writable through the DBI.
17–15 L1_Exit_Latency	L1 Exit Latency Writable through the DBI.
14–12 L0s_Exit_Latency	L0s Exit Latency Writable through the DBI.
11–10 Active_State_Link_PM_Support	Active State Link PM Support The default value is the value you specify during core configuration, writable through the DBI.
9–4 Maximum_Link_Width	Maximum Link Width Writable through the DBI.
Max_Link_Speeds	Max Link Speeds Indicates the supported maximum Link speeds of the associated Port. The encoding is the binary value of the bit location in the Supported Link Speeds Vector (in the Link Capabilities 2 register) that corresponds to the maximum Link speed. This field is writable through the DBI.

48.8.23 Link Control and Status Register (PCIE_RC_LCSR)

Offset: `CFG_PCIE_CAP + 0x10

Address: 1FF_C000h base + 80h offset = 1FF_C080h



PCIE_RC_LCSR field descriptions

Field	Description
31 Link_ Autonomous_ Bandwidth_ Status	Link Autonomous Bandwidth Status This bit is set by hardware to indicate that hardware has autonomously changed Link speed or width, without the Port transitioning through DL_Down status, for reasons other than to attempt to correct unreliable Link operation. This bit must be set if the Physical Layer reports a speed or a width change was initiated by the Downstream component that was indicated as an autonomous change. **NOTE:** This bit is not applicable and is reserved for Endpoints, PCI Express-to-PCI/PCI-X bridges.
30 Link_Bandwidth_ Management_ Status	Link Bandwidth Management Status This bit is set by hardware to indicate that either of the following has occurred without the Port transitioning through DL_Down status: •A Link retraining has completed following a write of 1b to the Retrain Link bit.

Table continues on the next page...

PCIE_RC_LCSR field descriptions (continued)

Field	Description
	•Hardware has changed Link speed or width to attempt to correct unreliable Link operation, either through an LTSSM timeout or a higher level process. This bit must be set if the Physical Layer reports a speed or width change was initiated by the Downstream component that was not indicated as an autonomous change. **NOTE:** : This bit is set following any write of 1b to the Retrain Link bit, including when the Link is in the process of retraining for some other reason. **NOTE:** This bit is not applicable and is reserved for Endpoints, PCI Express-to-PCI/PCI-X bridges.
29 Data_Link_ Layer_Active	Data Link Layer Active This bit must be implemented if the corresponding Data Link Layer Link Active Reporting capability bit is implemented. Otherwise, this bit must be hardwired to 0b.
28 Slot_Clock_ Configuration	Slot Clock Configuration Indicates that the component uses the same physical reference clock that the platform provides on the connector. The default value is the value you select during hardware configuration, writable through the DBI.
27 Link_Training	Link Training This bit is not applicable and is reserved for Endpoints, PCI Express to PCI/PCI-X bridges.
26 -	This field is reserved. Reserved Undefined for PCI Express 1.1 (Was Training Error for PCI Express 1.0a)
25–20 Negotiated_Link_ Width	Negotiated Link Width Set automatically by hardware after Link initialization. The value is undefined when link is not up.
19–16 Link_Speed	Link Speed Indicates the negotiated Link speed. The encoding is the binary value of the bit location in the Supported Link Speeds Vector (in the Link Capabilities 2 register) that corresponds to the current Link speed. Possible values are: 0001 Gen1 2.5 GT/s 0010 Gen2 5.0 GT/s
15–12 -	This field is reserved. Reserved
11 Link_ Autonomous_ Bandwidth_ Interrupt_Enable	Link Autonomous Bandwidth Interrupt Enable When set, this bit enables the generation of an interrupt to indicate that the Link Autonomous Bandwidth Status bit has been set. **NOTE:** This bit is not applicable and is reserved for Endpoints, PCI Express-to-PCI/PCI-X bridges.
10 Link_Bandwidth_ Management_ Interrupt_Enable	Link Bandwidth Management Interrupt Enable When set, this bit enables the generation of an interrupt to indicate that the Link Bandwidth Management Status bit has been set. **NOTE:** This bit is not applicable and is reserved for Endpoints, PCI Express-to-PCI/PCI-X bridges.

Table continues on the next page...

PCIE_RC_LCSR field descriptions (continued)

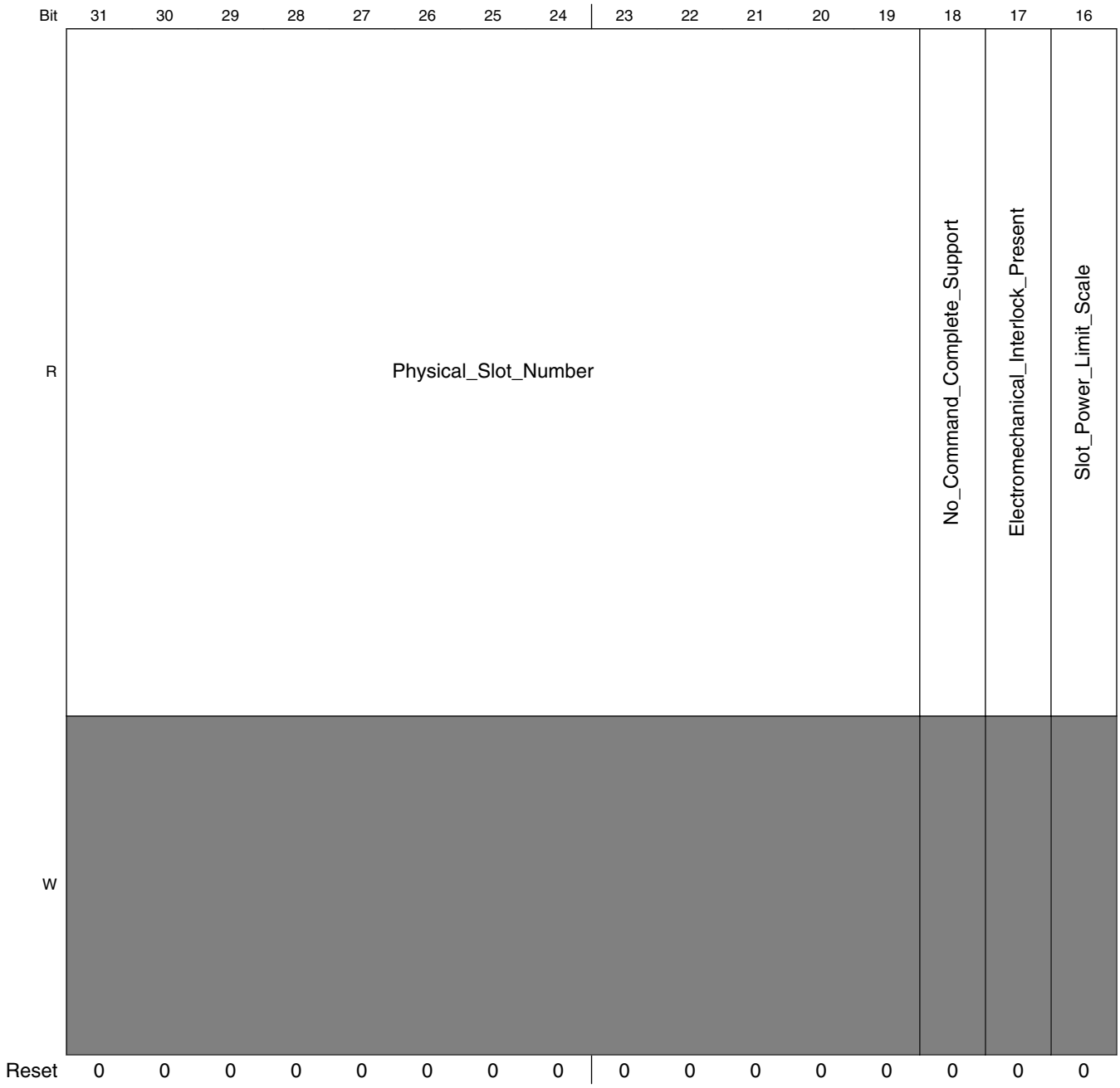
Field	Description
9 Hardware_ Autonomous_ Width_Disable	Hardware Autonomous Width Disable Not supported, hardwired to 0.
8 Enable_Clock_ Power_ Management	Enable Clock Power Management Hardwired to 0 if Clock Power Management is disabled in the Link Capabilities register.
7 Extended_Synch	Extended Synch
6 Common_Clock_ Configuration	Common Clock Configuration
5 Retrain_Link	Retrain Link This bit is reserved for PCI Express-to-PCI/PCI-X bridges.
4 Link_Disable	Link Disable This bit is reserved for PCI Express-to-PCI/PCI-X bridges.
3 RCB	Read Completion Boundary (RCB) RC: Writable through DBI
2 -	This field is reserved. Reserved
Active_State_ Link_PM_Control	Active State Link PM Control

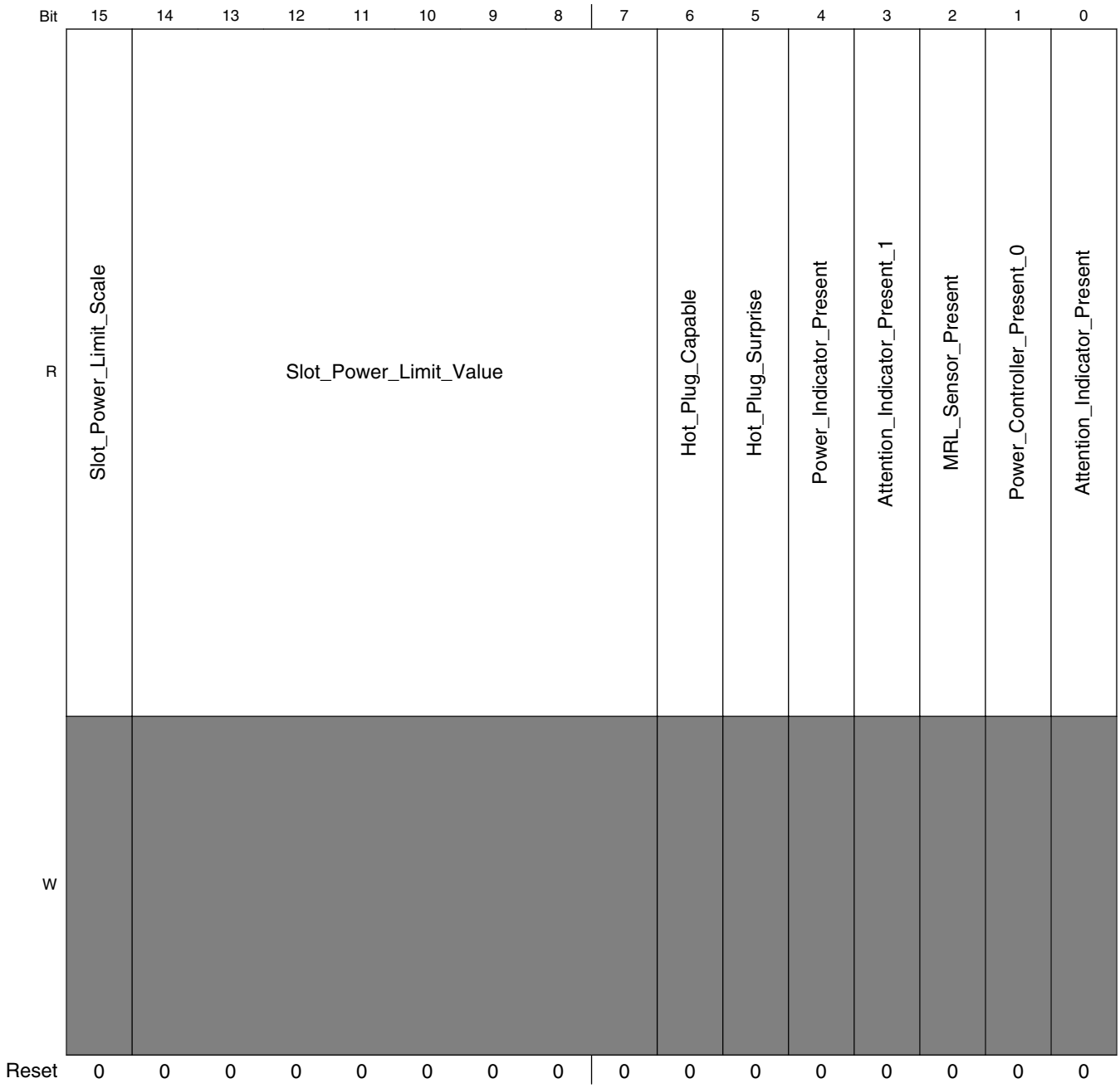
48.8.24 Slot Capabilities Register (PCIE_RC_SCR)

This section applies only to Downstream Ports (for example, RC.

Offset: `CFG_PCIE_CAP + 0x14

Address: 1FF_C000h base + 84h offset = 1FF_C084h





PCIE_RC_SCR field descriptions

Field	Description
31–19 Physical_Slot_Number	Physical Slot Number, writable through the DBI
18 No_Command_ Complete_Support	No Command Complete Support, writable through the DBI
17 Electromechanical_ Interlock_Present	Electromechanical Interlock Present, writable through the DBI

Table continues on the next page...

PCIE_RC_SCR field descriptions (continued)

Field	Description
16–15 Slot_Power_Limit_ Scale	Slot Power Limit Scale, writable through the DBI
14–7 Slot_Power_Limit_ Value	Slot Power Limit Value, writable through the DBI
6 Hot_Plug_Capable	Hot-Plug Capable, writable through the DBI
5 Hot_Plug_Surprise	Hot-Plug Surprise, writable through the DBI
4 Power_Indicator_ Present	Power Indicator Present, writable through the DBI
3 Attention_Indicator_ Present_1	Attention Indicator Present, writable through the DBI
2 MRL_Sensor_Present	MRL Sensor Present, writable through the DBI
1 Power_Controller_ Present_0	Power Controller Present, writable through the DBI
0 Attention_Indicator_ Present	Attention Indicator Present, writable through the DBI

48.8.25 Slot Control and Status Register (PCIE_RC_SCSR)

This section applies only to Downstream Ports (for example, RC.

Offset: `CFG_PCIE_CAP + 0x18

Address: 1FF_C000h base + 88h offset = 1FF_C088h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved							Data_Link_Layer_State_Changed	Electromechanical_Interlock_Status	Presence_Detect_State	MRL_Sensor_State	Command_Completed	Presence_Detect_Changed	MRL_Sensor_Changed	Power_Fault_Detected	Attention_Button_Pressed
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved			Data_Link_Layer_State_Changed_Enable	Electromechanical_Interlock_Control	Power_Controller_Control	Power_Indicator_Control		Attention_Indicator_Control		Hot_Plug_Interrupt_Enable	Command_Completed_Interrupt_Enable	Presence_Detect_Changed_Enable	MRL_Sensor_Changed_Enable	Power_Fault_Detected_Enable	Attention_Button_Pressed_Enable
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_RC_SCSR field descriptions

Field	Description
31–25 -	This field is reserved. Reserved
24 Data_Link_Layer_State_Changed	Data Link Layer State Changed
23 Electromechanical_Interlock_Status	Electromechanical Interlock Status
22 Presence_Detect_State	Presence Detect State
21 MRL_Sensor_State	MRL Sensor State
20 Command_Completed	Command Completed
19 Presence_Detect_Changed	Presence Detect Changed
18 MRL_Sensor_Changed	MRL Sensor Changed
17 Power_Fault_Detected	Power Fault Detected
16 Attention_Button_Pressed	Attention Button Pressed
15–13 -	This field is reserved. Reserved
12 Data_Link_Layer_State_Changed_Enable	Data Link Layer State Changed Enable
11 Electromechanical_Interlock_Control	Electromechanical Interlock Control

Table continues on the next page...

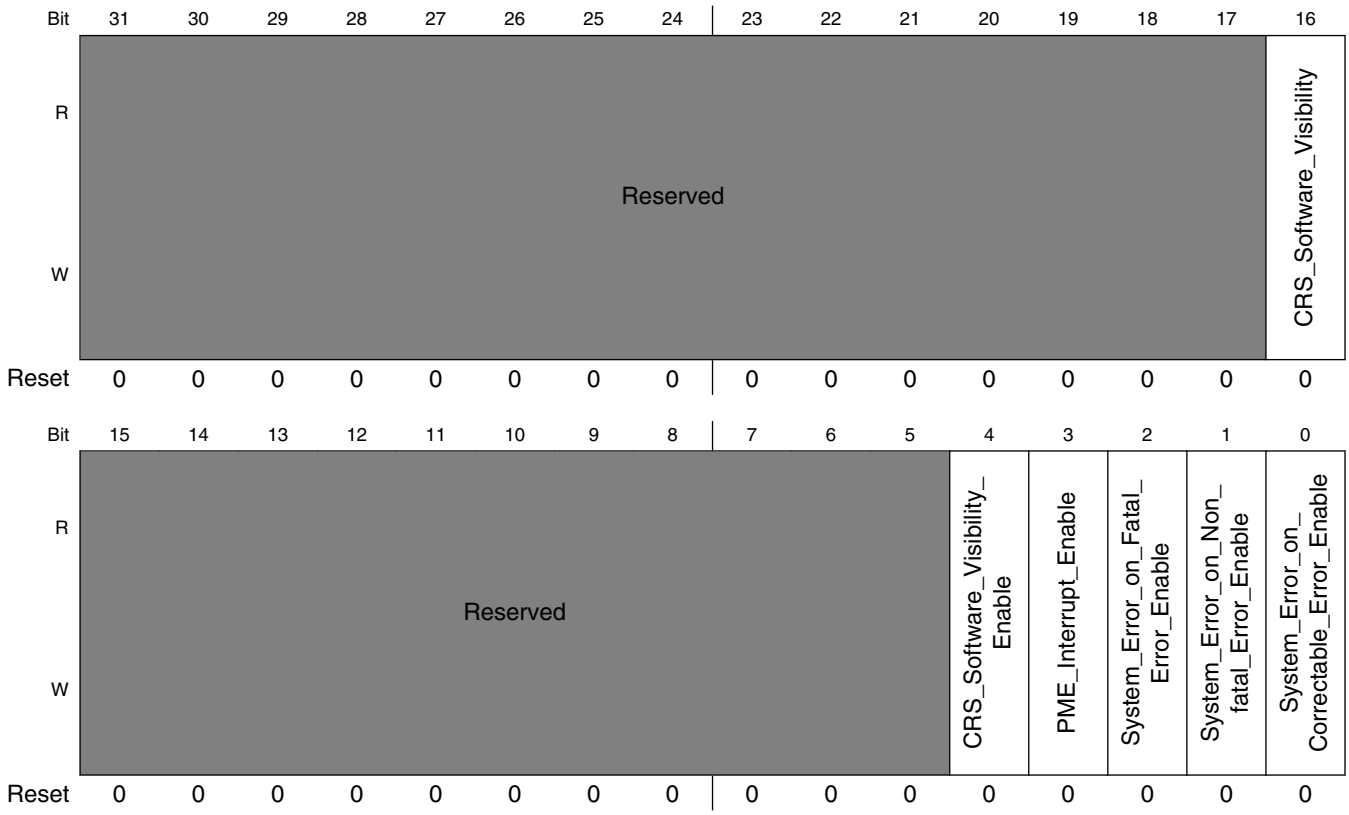
PCIE_RC_SCSR field descriptions (continued)

Field	Description
10 Power_Controller_ Control	Power Controller Control
9–8 Power_Indicator_ Control	Power Indicator Control
7–6 Attention_Indicator_ Control	Attention Indicator Control
5 Hot_Plug_Interrupt_ Enable	Hot-Plug Interrupt Enable
4 Command_ Completed_Interrupt_ Enable	Command Completed Interrupt Enable
3 Presence_Detect_ Changed_Enable	Presence Detect Changed Enable
2 MRL_Sensor_ Changed_Enable	MRL Sensor Changed Enable
1 Power_Fault_ Detected_Enable	Power Fault Detected Enable
0 Attention_Button_ Pressed_Enable	Attention Button Pressed Enable

48.8.26 Root Control and Capabilities Register (PCIE_RC_RCCR)

Offset: `CFG_PCIE_CAP + 0x1C

Address: 1FF_C000h base + 8Ch offset = 1FF_C08Ch



PCIE_RC_RCCR field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 CRS_Software_Visibility	CRS Software Visibility Not supported, hardwired to 0x0.
15–5 -	This field is reserved. Reserved
4 CRS_Software_Visibility_Enable	CRS Software Visibility Enable Not supported, hardwired to 0x0.
3 PME_Interrupt_Enable	PME Interrupt Enable

Table continues on the next page...

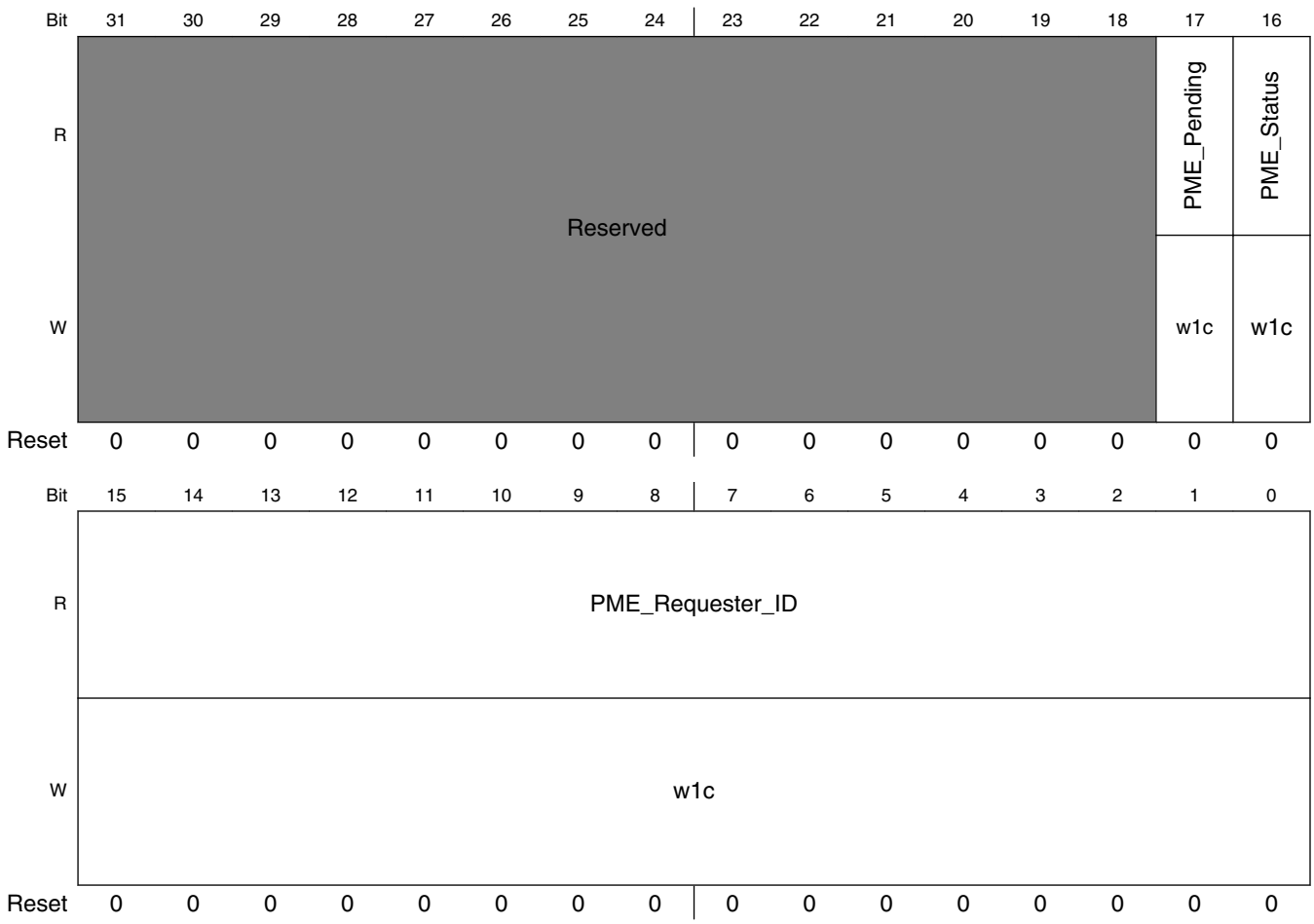
PCIE_RC_RCCR field descriptions (continued)

Field	Description
2 System_Error_ on_Fatal_Error_ Enable	System Error on Fatal Error Enable
1 System_Error_ on_Non_fatal_ Error_Enable	System Error on Non-fatal Error Enable
0 System_Error_ on_Correctable_ Error_Enable	System Error on Correctable Error Enable

48.8.27 Root Status Register (PCIE_RC_RSR)

Offset: `CFG_PCIE_CAP + 0x20

Address: 1FF_C000h base + 90h offset = 1FF_C090h



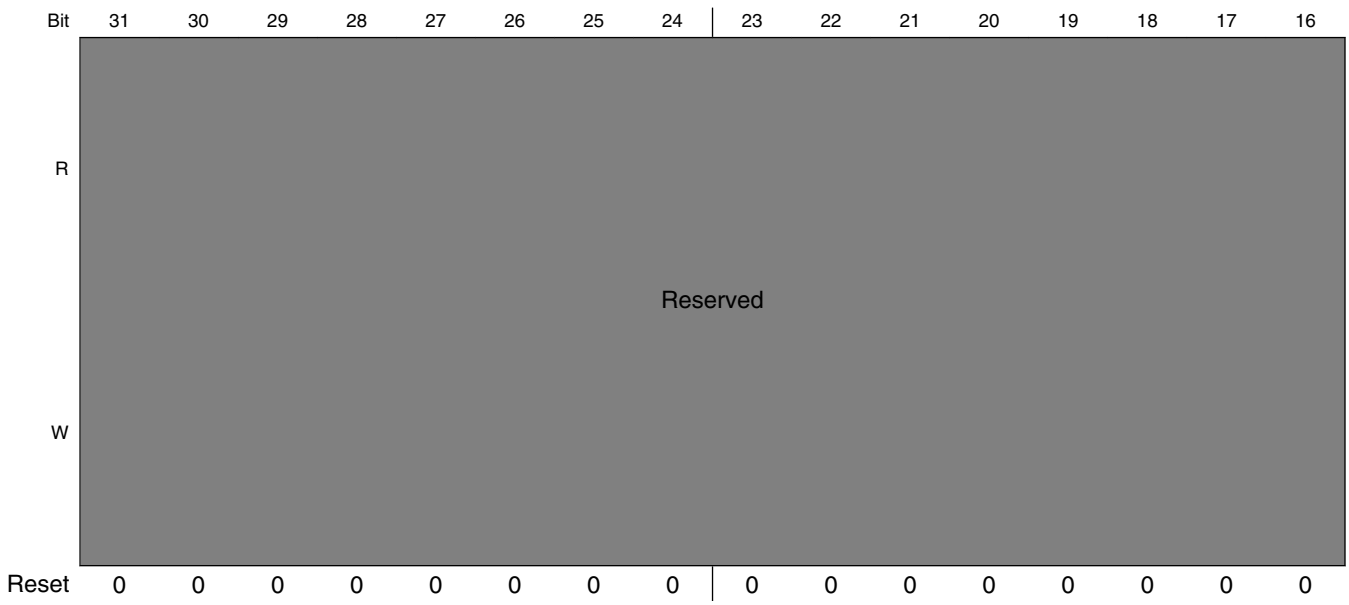
PCIE_RC_RSR field descriptions

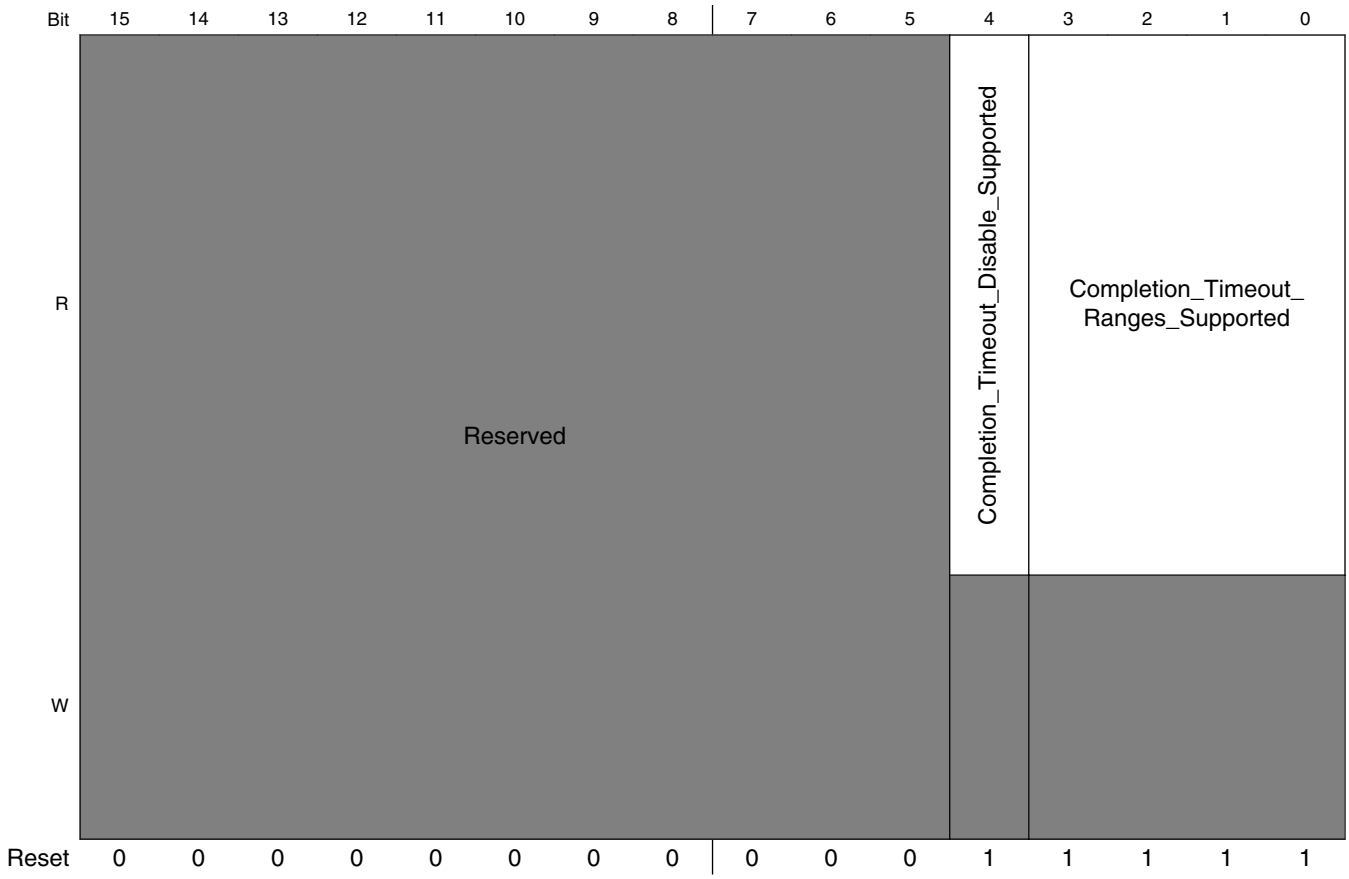
Field	Description
31–18 -	This field is reserved. Reserved
17 PME_Pending	PME Pending
16 PME_Status	PME Status
PME_Requester_ID	PME Requester ID

48.8.28 Device Capabilities 2 Register (PCIE_RC_DCR2)

Offset: `CFG_PCIE_CAP + 0x24

Address: 1FF_C000h base + 94h offset = 1FF_C094h





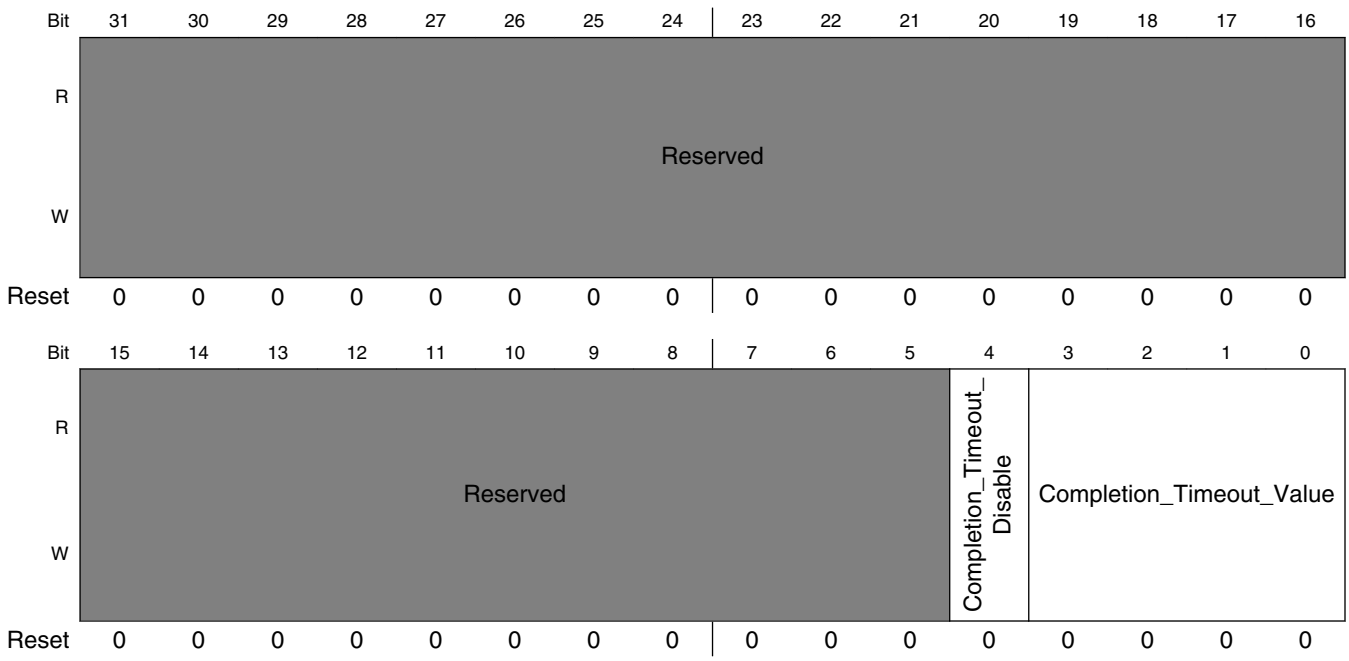
PCIE_RC_DCR2 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 Completion_ Timeout_ Disable_ Supported	Completion Timeout Disable Supported
Completion_ Timeout_ Ranges_ Supported	Completion Timeout Ranges Supported This field is applicable only to Root Ports, Endpoints that issue Requests on their own behalf, and PCI Express to PCI/PCI-X Bridges that take ownership of Requests issued on PCI Express. the default value is 0xf (A, B, C and D ranges supported)

48.8.29 Device Control and Status 2 Register (PCIE_RC_DCSR2)

Offset: `CFG_PCIE_CAP + 0x28

Address: 1FF_C000h base + 98h offset = 1FF_C098h



PCIE_RC_DCSR2 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 Completion_ Timeout_Disable	Completion Timeout Disable
Completion_ Timeout_Value	Completion Timeout Value If the default range is chosen, the core will have a timeout in the range of 16ms to 55ms. . following encodings apply: Values not defined below are reserved. 0000 Default range: 50 ìs to 50 ms 0001 50 ìs to 100 ìs 0010 1 ms to 10 ms 0101 16 ms to 55 ms 0110 65 ms to 210 ms 1001 260 ms to 900 ms

Table continues on the next page...

PCIE_RC_DCSR2 field descriptions (continued)

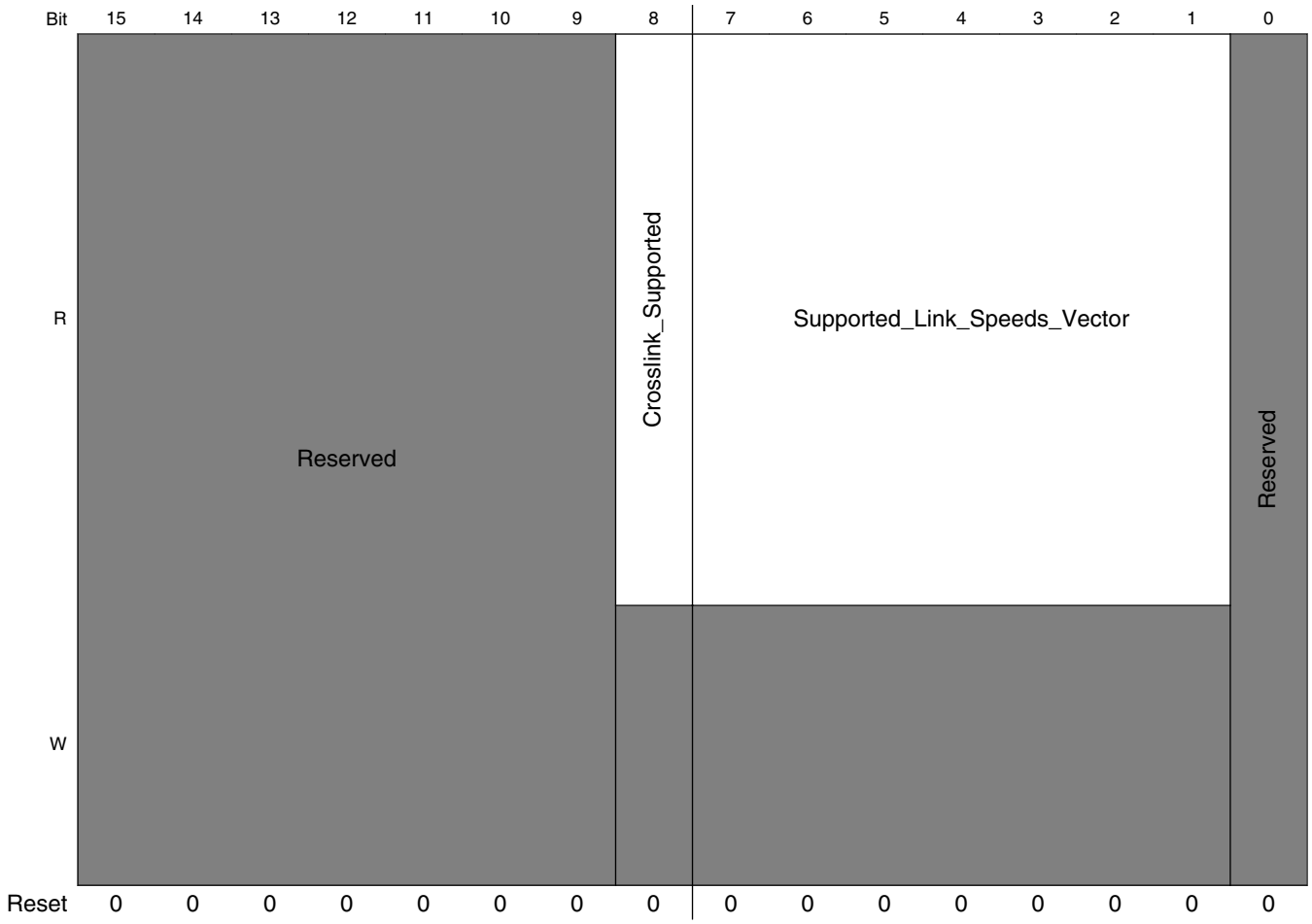
Field	Description
	1010 1 s to 3.5 s 1101 4 s to 13 s 1110 17 s to 64 s

48.8.30 Link Capabilities 2 Register (PCIE_RC_LCR2)

Offset: `CFG_PCIE_CAP + 0x2C

Address: 1FF_C000h base + 9Ch offset = 1FF_C09Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



PCIE_RC_LCR2 field descriptions

Field	Description
31–9 -	This field is reserved. Reserved
8 Crosslink_ Supported	Crosslink Supported
7–1 Supported_Link_ Speeds_Vector	Supported Link Speeds Vector Indicates the supported Link speeds of the associated Port. For each bit, a value of 1b indicates that the corresponding Link speed is supported; otherwise, the Link speed is not supported. Bit definitions are: Bit 1 2.5 GT/s Bit 2 5.0 GT/s Bit 3 8.0 GT/s Bits 7:4 reserved This field is writable through the DBI.
0 -	This field is reserved. Reserved

48.8.31 Link Control and Status 2 Register (PCIE_RC_LCSR2)

Offset: `CFG_PCIE_CAP + 30

Address: 1FF_C000h base + A0h offset = 1FF_C0A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved										Link_Equalization_Request	Equalization_Phase_3_Successful	Equalization_Phase_2_Successful	Equalization_Phase_1_Successful	Equalization_Complete	Current_Deemphasis_Level
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Compliance_Pre_set_Deemphasis				Compliance_SOS	Enter_Modified_Compliance	Transmit_Margin			Selectable_Deemphasis	Hardware_Autonomous_Speed_Disable	Enter_Compliance	Target_Link_Speed			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_RC_LCSR2 field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21 Link_ Equalization_ Request	Link Equalization Request
20 Equalization_ Phase_3_ Successful	Equalization Phase 3 Successful
19 Equalization_ Phase_2_ Successful	Equalization Phase 2 Successful

Table continues on the next page...

PCIE_RC_LCSR2 field descriptions (continued)

Field	Description
18 Equalization_ Phase_1_ Successful	Equalization Phase 1 Successful
17 Equalization_ Complete	Equalization Complete
16 Current_ Deemphasis_ Level	Current De-emphasis Level
15–12 Compliance_ Pre_set_ Deemphasis	Compliance Pre-set/ De-emphasis
11 Compliance_ SOS	Compliance SOS When set to 1b, the LTSSM is required to send SKP Ordered Sets periodically in between the (modified) compliance patterns. GT/s speed are permitted to hardwire this bit to 0b. **NOTE:** When the Link is operating at 2.5 GT/s, the setting of this bit has no effect. Components that support only 2.5
10 Enter_Modified_ Compliance	Enter Modified Compliance When this bit is set to 1b, the device transmits modified compliance pattern if the LTSSM enters Polling. Compliance state.
9–7 Transmit_Margin	Transmit Margin This field is reset to 000b on entry to the LTSSM Polling. Compliance substate. Components that support only the 2.5 GT/s speed are permitted to hard-wire this bit to 0b. When operating in 5.0 GT/s mode with full swing, the de-emphasis ratio must be maintained within +/- 1 dB from the specification-defined operational value (either -3.5 or -6 dB). This field controls the value of the non-de-emphasized voltage level at the Transmitter pins: 000 800-1200 mV for full swing 400-600 mV for half- swing 001-010 values must be monotonic with a non-zero slope 011 200-400 mV for full-swing and 100-200 mV for halfswing 100-111 reserved
6 Selectable_ Deemphasis	Selectable De-emphasis When the Link is operating at 2.5 GT/s speed, the setting of this bit has no effect. Components that support only the 2.5 GT/s speed are permitted to hardwire this bit to 0b. Default value is implementation-specific, unless a specific value is required for a selected form factor or platform. When the Link is operating at 5.0 GT/s speed, selects the level of de-emphasis: 1 -3.5 dB 0 -6 dB

Table continues on the next page...

PCIE_RC_LCSR2 field descriptions (continued)

Field	Description
5 Hardware_ Autonomous_ Speed_Disable	Hardware Autonomous Speed Disable When cfg_hw_auto_sp_dis signal is asserted, the application must disable hardware from changing the Link speed for device-specific reasons other than attempting to correct unreliable Link operation by reducing Link speed. Initial transition to the highest supported common link speed is not blocked by this signal.
4 Enter_ Compliance	Enter Compliance Software is permitted to force a link to enter Compliance mode at the speed indicated in the Target Link Speed field by setting this bit to 1b in both components on a link and then initiating a hot reset on the link. The default value of this field following Fundamental Reset is 0b.
Target_Link_ Speed	Target Link Speed For Downstream ports, this field sets an upper limit on link operational speed by restricting the values advertised by the upstream component in its training sequences: The encoding is the binary value of the bit in the Supported Link Speeds Vector (in the Link Capabilities 2 register) that corresponds to the desired target Link speed. **NOTE:** If a value is written to this field that does not correspond to a speed included in the Supported Link Speeds field, the result is undefined. **NOTE:** The default value of this field is the highest link speed supported by the component (as reported in the Max Link Speed field of the Link Capabilities Register) unless the corresponding platform / form factor requires a different default value. **NOTE:** Components that support only the 2.5 GT/s speed are permitted to hardwire this field to 0000b. All other encodings are reserved. 0000001 Gen1 2.5 GT/s 0000010 Gen2 5.0 GT/s

48.8.32 AER Capability Header (PCIE_RC_AER)

The core implements the following PCI Express Extended Capabilities registers:

Advanced Error Reporting Capability register set

Virtual Channel Capability register set

Address: 0x100

Address: 1FF_C000h base + 100h offset = 1FF_C100h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Next_Capability_Offset												Capability_ Version				PCI_Express_Extended_Capability_ID															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

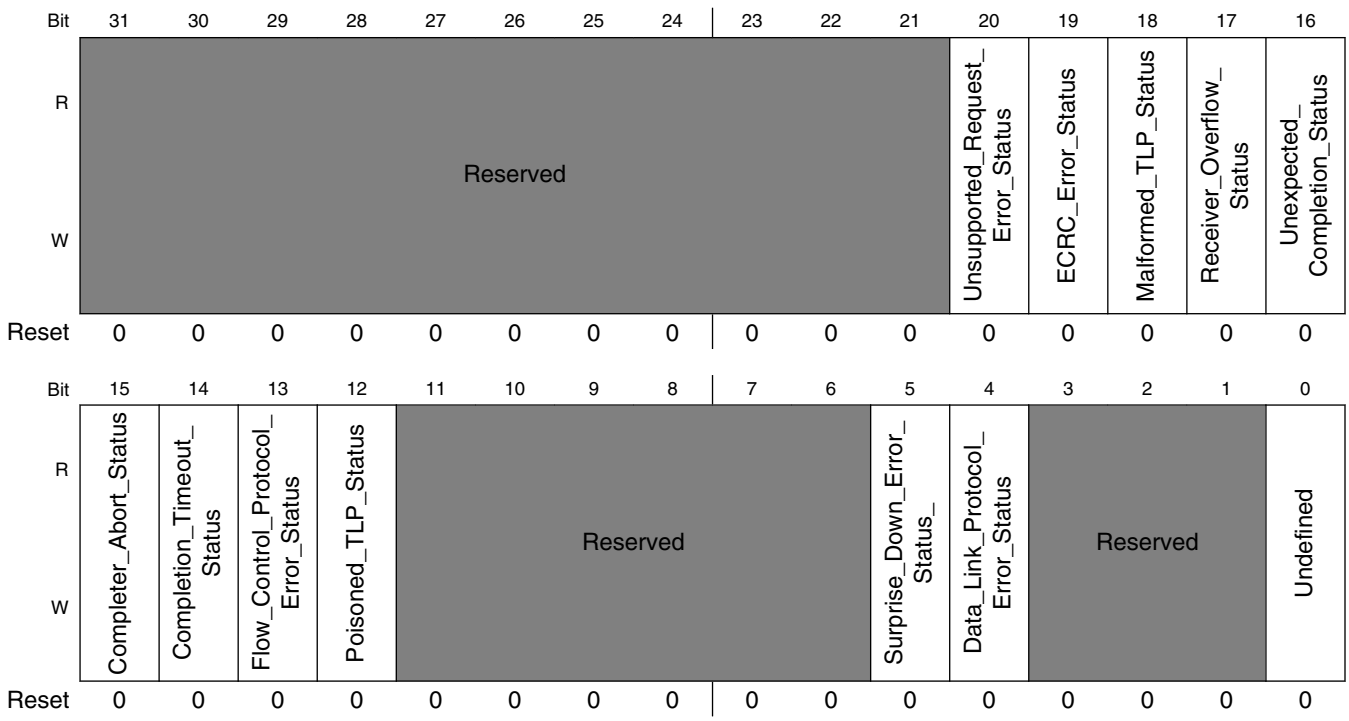
PCIE_RC_AER field descriptions

Field	Description
31–20 Next_Capability_Offset	Next Capability Offset
19–16 Capability_Version	Capability Version
PCI_Express_Extended_Capability_ID	PCI Express Extended Capability ID Value is 0x1 for Advanced Error Reporting.

48.8.33 Uncorrectable Error Status Register (PCIE_RC_UESR)

Offset: 0x04

Address: 1FF_C000h base + 104h offset = 1FF_C104h



PCIE_RC_UESR field descriptions

Field	Description
31–21 -	This field is reserved. Reserved

Table continues on the next page...

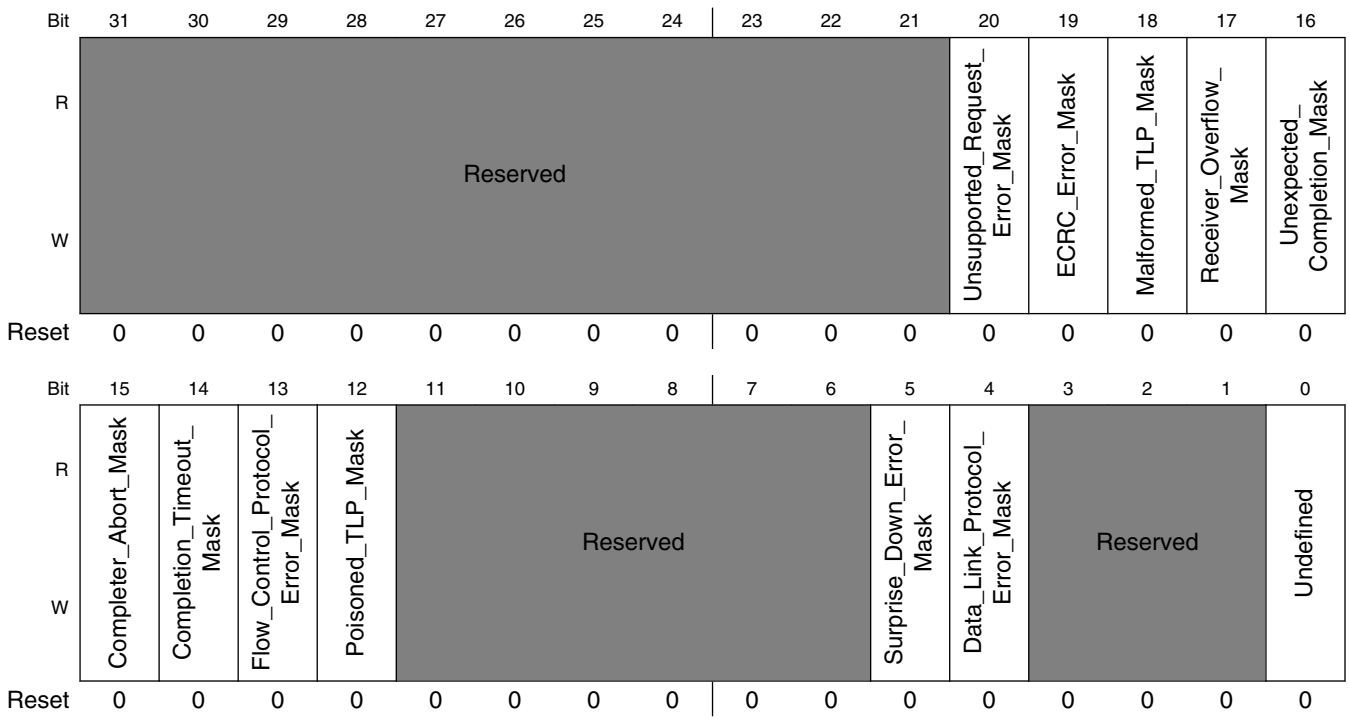
PCIE_RC_UESR field descriptions (continued)

Field	Description
20 Unsupported_ Request_Error_ Status	Unsupported Request Error Status
19 ECRC_Error_ Status	ECRC Error Status
18 Malformed_TLP_ Status	Malformed TLP Status
17 Receiver_ Overflow_Status	Receiver Overflow Status
16 Unexpected_ Completion_ Status	Unexpected Completion Status
15 Completer_ Abort_Status	Completer Abort Status
14 Completion_ Timeout_Status	Completion Timeout Status
13 Flow_Control_ Protocol_Error_ Status	Flow Control Protocol Error Status
12 Poisoned_TLP_ Status	Poisoned TLP Status
11–6 -	This field is reserved. Reserved
5 Surprise_Down_ Error_Status_	Surprise Down Error Status (not supported)
4 Data_Link_ Protocol_Error_ Status	Data Link Protocol Error Status
3–1 -	This field is reserved. Reserved
0 Undefined	Undefined for PCI Express 1.1 (Was Training Error Status for PCI Express 1.0a)

48.8.34 Uncorrectable Error Mask Register (PCIE_RC_UEMR)

Offset: 0x08

Address: 1FF_C000h base + 108h offset = 1FF_C108h



PCIE_RC_UEMR field descriptions

Field	Description
31–21 -	This field is reserved. Reserved
20 Unsupported_Request_Error_Mask	Unsupported Request Error Mask
19 ECRC_Error_Mask	ECRC Error Mask
18 Malformed_TLP_Mask	Malformed TLP Mask
17 Receiver_Overflow_Mask	Receiver Overflow Mask

Table continues on the next page...

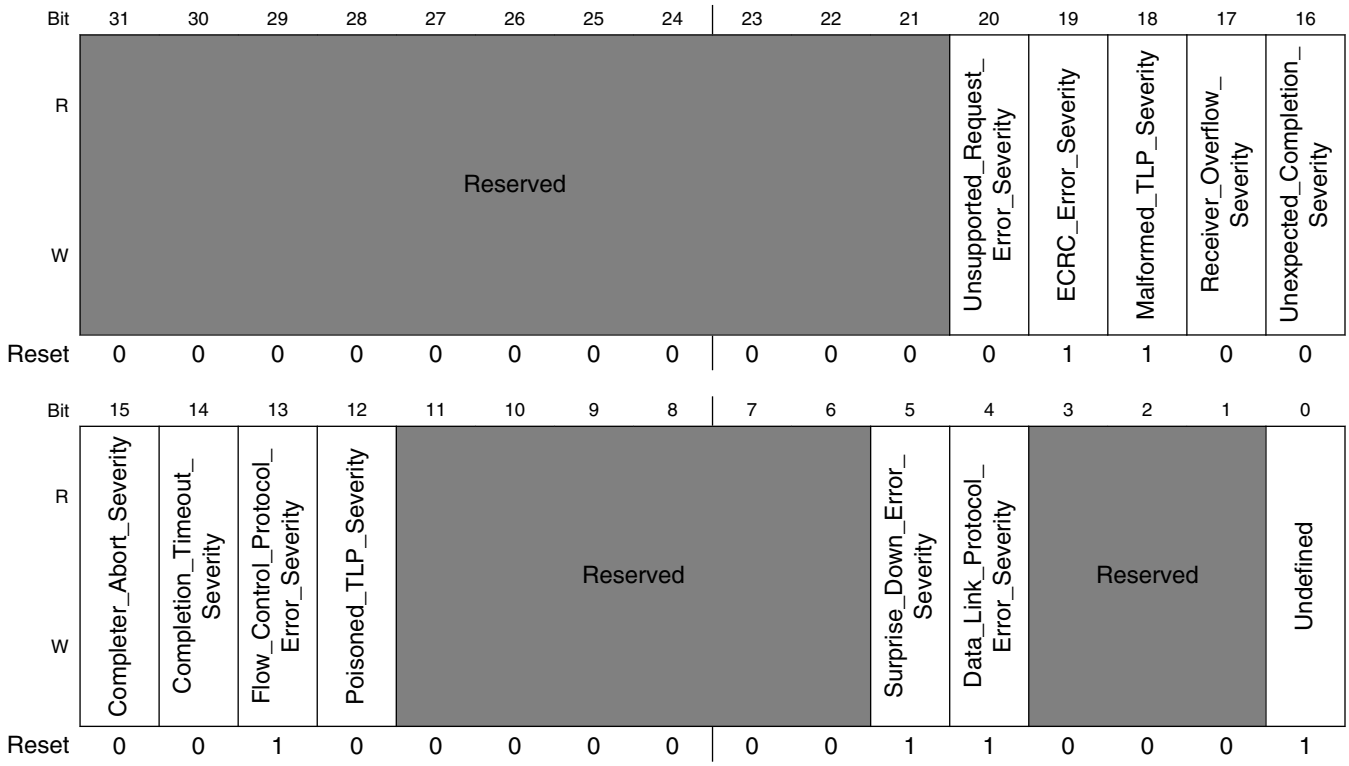
PCIE_RC_UEMR field descriptions (continued)

Field	Description
16 Unexpected_ Completion_ Mask	Unexpected Completion Mask
15 Completer_ Abort_Mask	Completer Abort Mask
14 Completion_ Timeout_Mask	Completion Timeout Mask
13 Flow_Control_ Protocol_Error_ Mask	Flow Control Protocol Error Mask
12 Poisoned_TLP_ Mask	Poisoned TLP Mask
11–6 -	This field is reserved. Reserved
5 Surprise_Down_ Error_Mask	Surprise Down Error Mask (not supported)
4 Data_Link_ Protocol_Error_ Mask	Data Link Protocol Error Mask
3–1 -	This field is reserved. Reserved
0 Undefined	Undefined for PCI Express 1.1 (Was Training Error Mask for PCI Express 1.0a)

48.8.35 Uncorrectable Error Severity Register (PCIE_RC_UESevR)

Offset: 0x0C

Address: 1FF_C000h base + 10Ch offset = 1FF_C10Ch



PCIE_RC_UESevR field descriptions

Field	Description
31–21 -	This field is reserved. Reserved
20 Unsupported_Request_Error_Severity	Unsupported Request Error Severity
19 ECRC_Error_Severity	ECRC Error Severity
18 Malformed_TLP_Severity	Malformed TLP Severity

Table continues on the next page...

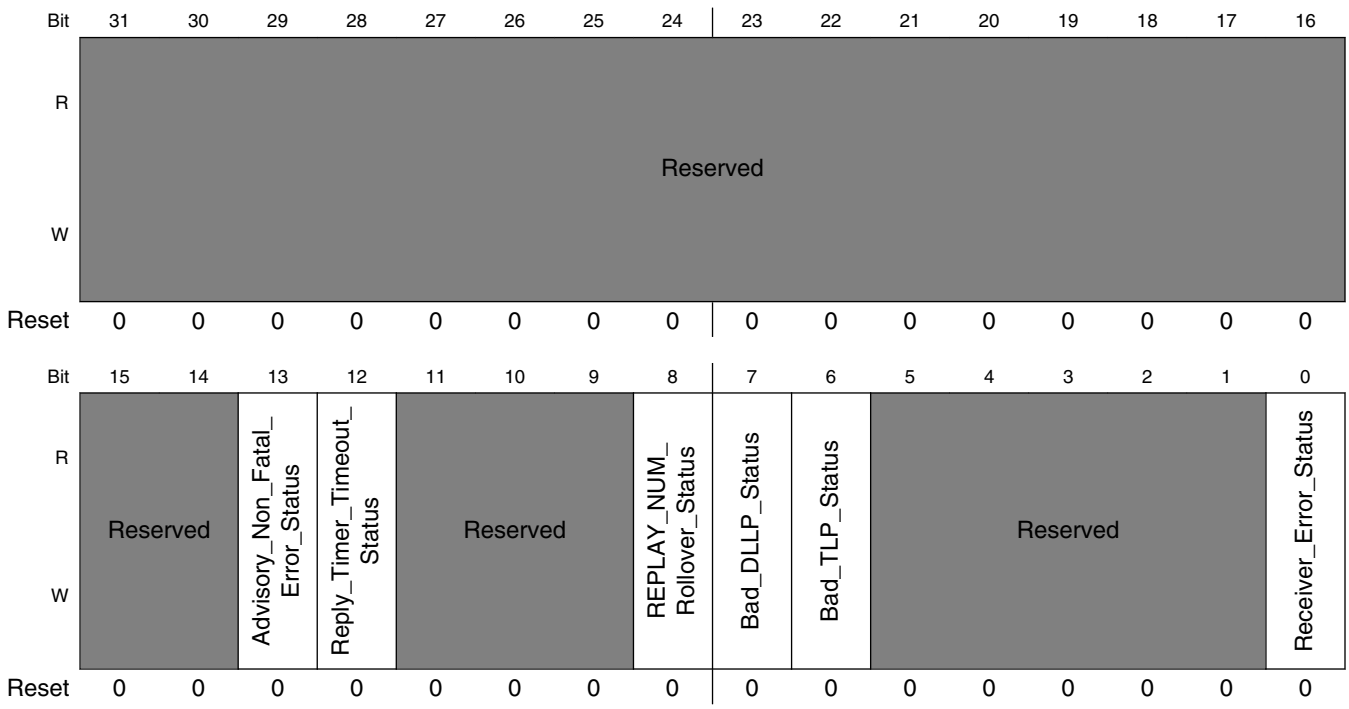
PCIE_RC_UESevR field descriptions (continued)

Field	Description
17 Receiver_ Overflow_ Severity	Receiver Overflow Severity
16 Unexpected_ Completion_ Severity	Unexpected Completion Severity
15 Completer_ Abort_Severity	Completer Abort Severity
14 Completion_ Timeout_Severity	Completion Timeout Severity
13 Flow_Control_ Protocol_Error_ Severity	Flow Control Protocol Error Severity
12 Poisoned_TLP_ Severity	Poisoned TLP Severity
11–6 -	This field is reserved. Reserved
5 Surprise_Down_ Error_Severity	Surprise Down Error Severity (not supported)
4 Data_Link_ Protocol_Error_ Severity	Data Link Protocol Error Severity
3–1 -	This field is reserved. Reserved
0 Undefined	Undefined for PCI Express 1.1 (Was Training Error Severity for PCI Express 1.0a)

48.8.36 Correctable Error Status Register (PCIE_RC_CESR)

Offset: 0x10

Address: 1FF_C000h base + 110h offset = 1FF_C110h



PCIE_RC_CESR field descriptions

Field	Description
31–14 -	This field is reserved. Reserved
13 Advisory_Non_Fatal_Error_Status	Advisory Non-Fatal Error Status
12 Reply_Timer_Timeout_Status	Reply Timer Timeout Status
11–9 -	This field is reserved. Reserved
8 REPLAY_NUM_Rollover_Status	REPLAY_NUM Rollover Status
7 Bad_DLLP_Status	Bad DLLP Status

Table continues on the next page...

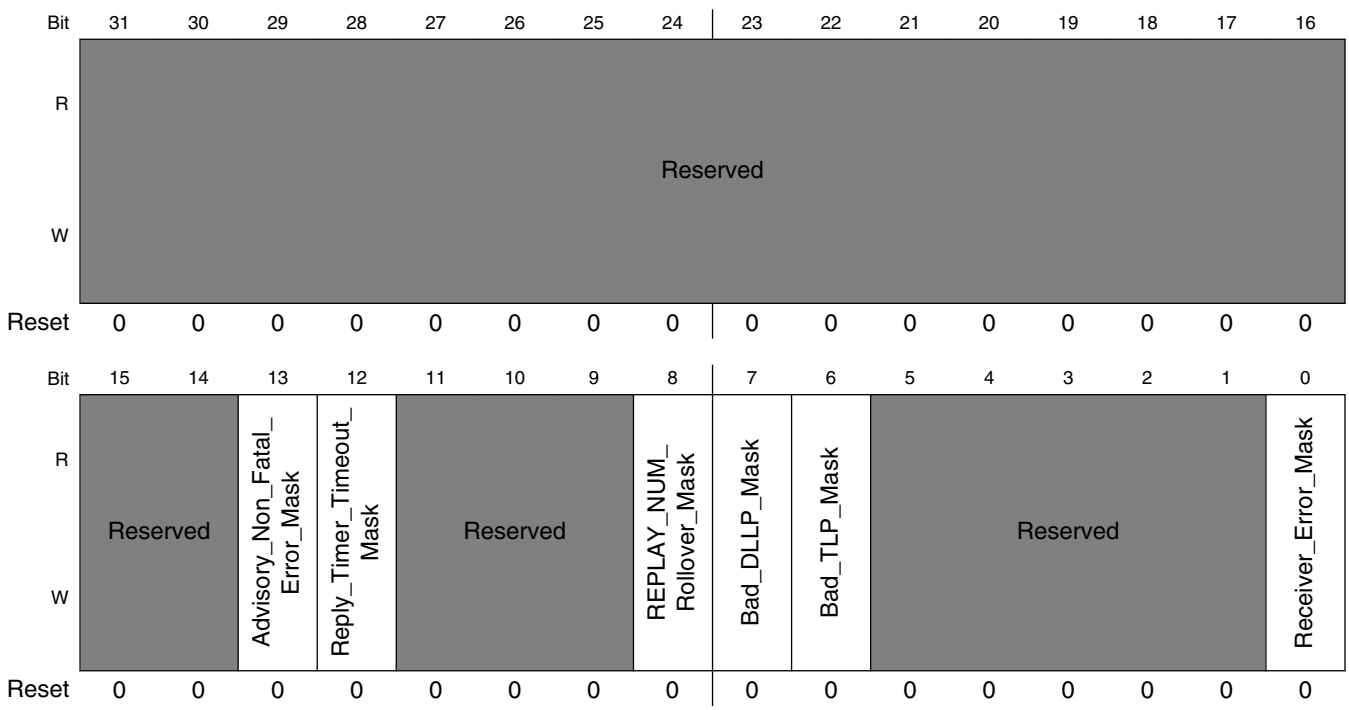
PCIE_RC_CESR field descriptions (continued)

Field	Description
6 Bad_TLP_Status	Bad TLP Status
5–1 -	This field is reserved. Reserved
0 Receiver_Error_Status	Receiver Error Status

48.8.37 Correctable Error Mask Register (PCIE_RC_CEMR)

Offset: 0x14

Address: 1FF_C000h base + 114h offset = 1FF_C114h



PCIE_RC_CEMR field descriptions

Field	Description
31–14 -	This field is reserved. Reserved

Table continues on the next page...

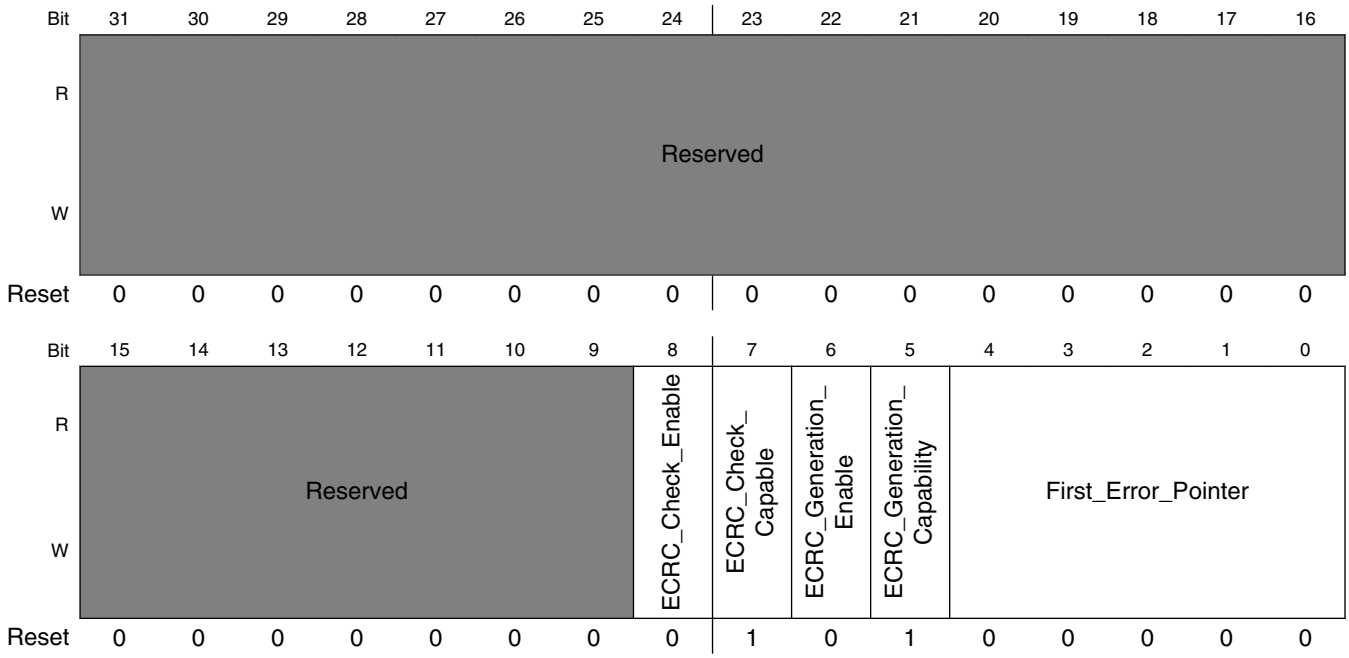
PCIE_RC_CEMR field descriptions (continued)

Field	Description
13 Advisory_Non_Fatal_Error_Mask	Advisory Non-Fatal Error Mask
12 Reply_Timer_Timeout_Mask	Reply Timer Timeout Mask
11–9 -	This field is reserved. Reserved
8 REPLAY_NUM_Rollover_Mask	REPLAY_NUM Rollover Mask
7 Bad_DLLP_Mask	Bad DLLP Mask
6 Bad_TLP_Mask	Bad TLP Mask
5–1 -	This field is reserved. Reserved
0 Receiver_Error_Mask	Receiver Error Mask

48.8.38 Advanced Capabilities and Control Register (PCIE_RC_ACCR)

Offset: 0x18

Address: 1FF_C000h base + 118h offset = 1FF_C118h



PCIE_RC_ACCR field descriptions

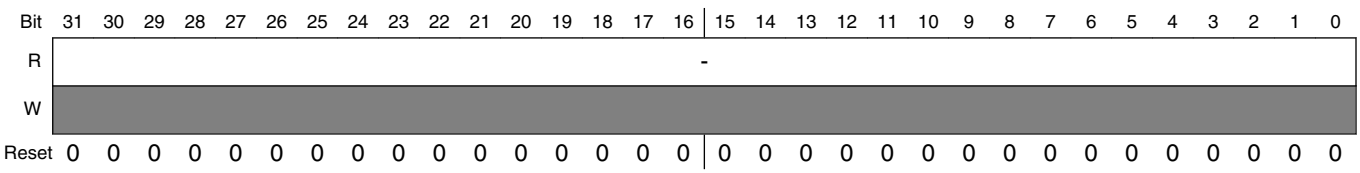
Field	Description
31–9 -	This field is reserved. Reserved
8 ECRC_Check_ Enable	ECRC Check Enable
7 ECRC_Check_ Capable	ECRC Check Capable
6 ECRC_ Generation_ Enable	ECRC Generation Enable
5 ECRC_ Generation_ Capability	ECRC Generation Capability
First_Error_ Pointer	First Error Pointer

48.8.39 Header Log Register (PCIE_RC_HLR)

Offset: 0x1C

The Header Log registers collect the header for the TLP corresponding to a detected error. See the PCI Express 3.0 Specification for details. Each of the Header Log registers is type ROS; the default reset value of each Header Log register is 0x00000000.

Address: 1FF_C000h base + 11Ch offset = 1FF_C11Ch



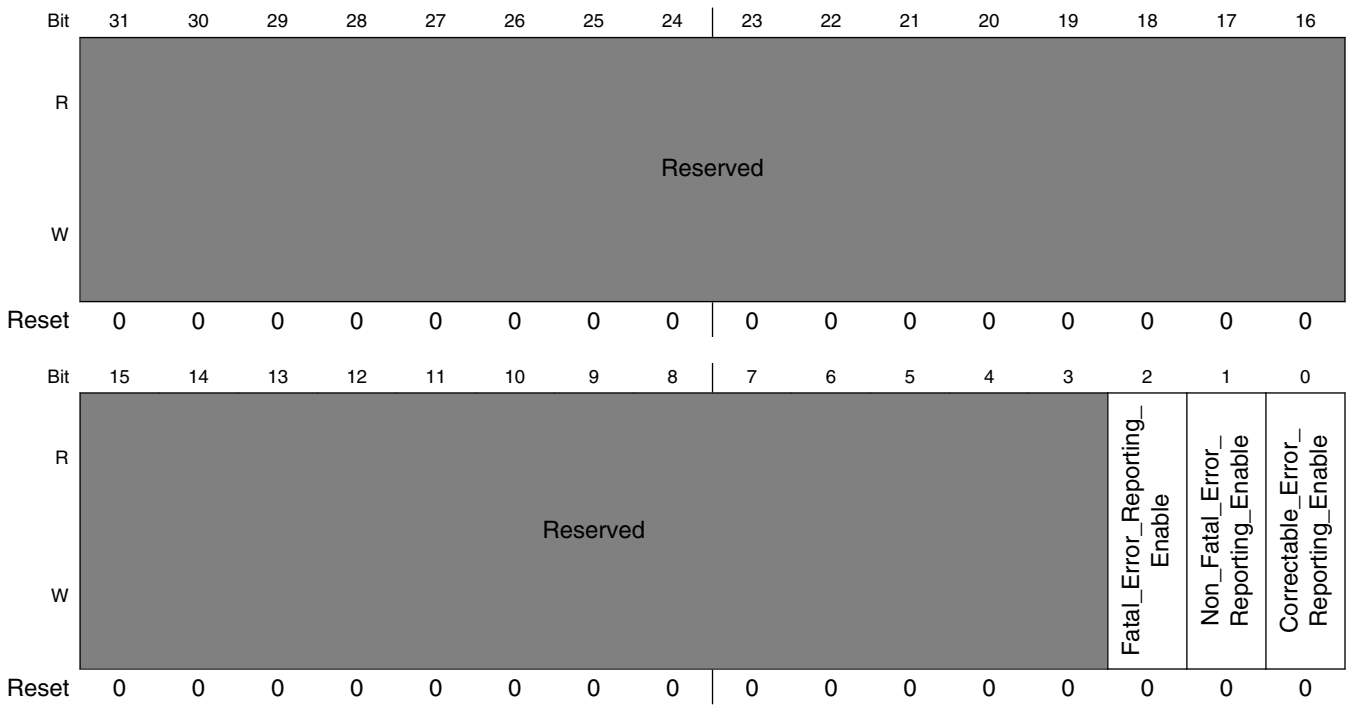
PCIE_RC_HLR field descriptions

Field	Description
-	Header Log Register (nth DWORD)

48.8.40 Root Error Command Register (PCIE_RC_RECR)

Offset: 0x100 + 0x2C

Address: 1FF_C000h base + 12Ch offset = 1FF_C12Ch



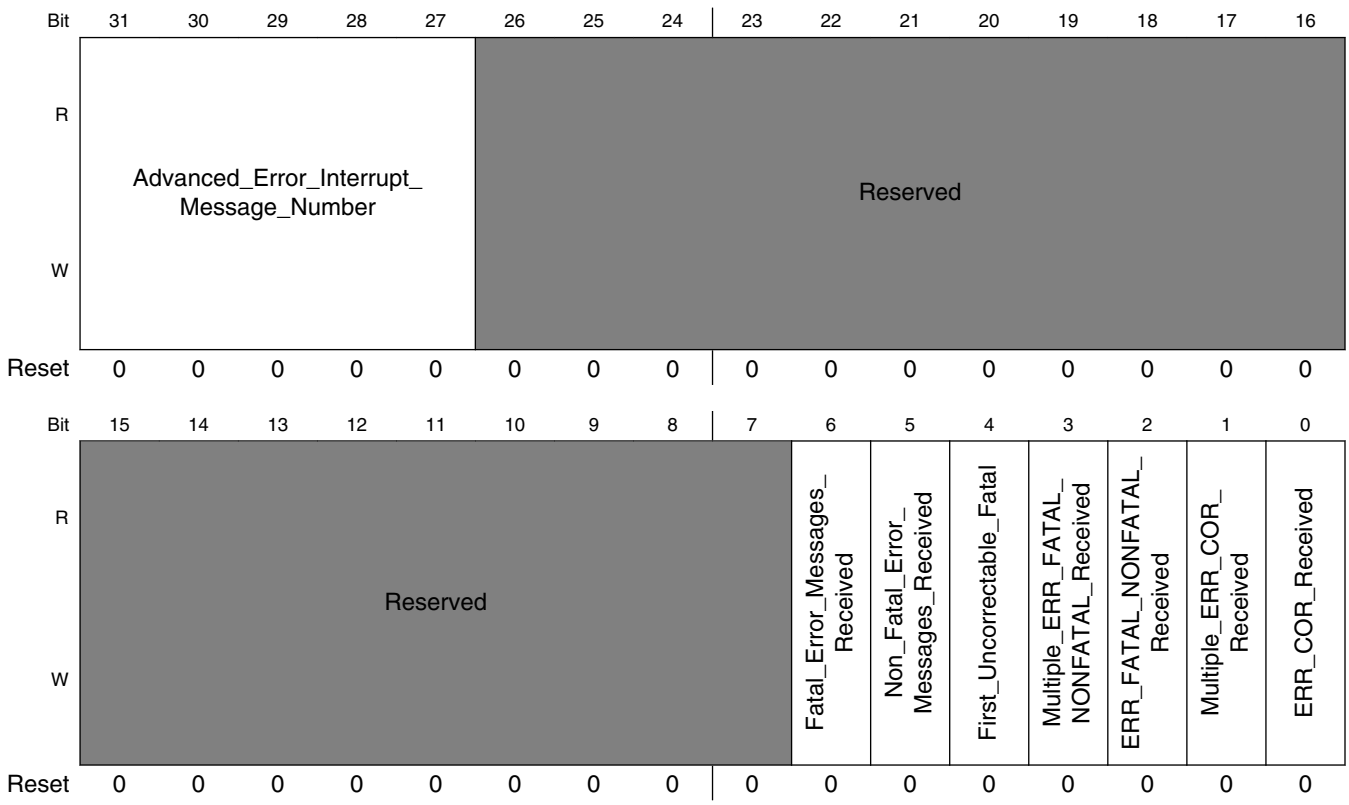
PCIE_RC_RECR field descriptions

Field	Description
31–3 -	This field is reserved. Reserved
2 Fatal_Error_ Reporting_ Enable	Fatal Error Reporting Enable
1 Non_Fatal_ Error_Reporting_ Enable	Non-Fatal Error Reporting Enable
0 Correctable_ Error_Reporting_ Enable	Correctable Error Reporting Enable

48.8.41 Root Error Status Register (PCIE_RC_RESR)

Offset: 0x100 + 0x30

Address: 1FF_C000h base + 130h offset = 1FF_C130h



PCIE_RC_RESR field descriptions

Field	Description
31–27 Advanced_Error_ Interrupt_ Message_ Number	Advanced Error Interrupt Message Number, writable through the DBI
26–7 -	This field is reserved. Reserved
6 Fatal_Error_ Messages_ Received	Fatal Error Messages Received
5 Non_Fatal_Error_ Messages_ Received	Non-Fatal Error Messages Received

Table continues on the next page...

PCIE_RC_RESR field descriptions (continued)

Field	Description
4 First_ Uncorrectable_ Fatal	First Uncorrectable Fatal
3 Multiple_ERR_ FATAL_ NONFATAL_ Received	Multiple ERR_FATAL/NONFATAL Received
2 ERR_FATAL_ NONFATAL_ Received	ERR_FATAL/NONFATAL Received
1 Multiple_ERR_ COR_Received	Multiple ERR_COR Received
0 ERR_COR_ Received	ERR_COR Received

48.8.42 Error Source Identification Register (PCIE_RC_ESIR)

Offset: 0x100 + 0x34

Address: 1FF_C000h base + 134h offset = 1FF_C134h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ERR_FATAL_NONFATAL_SID																ERR_COR_SID															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_RC_ESIR field descriptions

Field	Description
31–16 ERR_FATAL_ NONFATAL_SID	ERR_FATAL/NONFATAL Source Identification
ERR_COR_SID	ERR_COR Source Identification

48.8.43 VC Extended Capability Header (PCIE_RC_VCECHR)

Offset: 0x140

Address: 1FF_C000h base + 140h offset = 1FF_C140h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Next_Capability_Offset												Capability_ Version				Extended_Capability															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0

PCIE_RC_VCECHR field descriptions

Field	Description
31–20 Next_Capability_ Offset	Next Capability Offset
19–16 Capability_ Version	Capability Version
Extended_ Capability	PCI Express Extended Capability The default value is 0x2 for VC Capability.

48.8.44 Port VC Capability Register 1 (PCIE_RC_PVCCR1)

Offset: 0x140 + 0x4

Address: 1FF_C000h base + 144h offset = 1FF_C144h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				Port_ Arbitration_ Table_Entry_ Size		Reference_ Clock		Reserved	Low_Priority_ Extended_VC_ Count			Reserved	Extended_VC_ Count		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

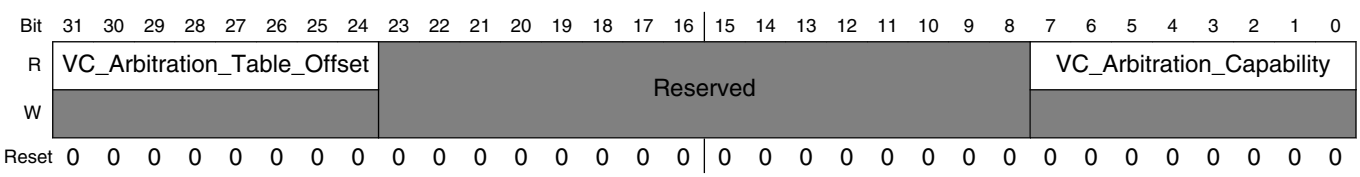
PCIE_RC_PVCCR1 field descriptions

Field	Description
31–12 -	This field is reserved. Reserved
11–10 Port_Arbitration_Table_Entry_Size	Port Arbitration Table Entry Size
9–8 Reference_Clock	Reference Clock
7 -	This field is reserved. Reserved
6–4 Low_Priority_Extended_VC_Count	Low Priority Extended VC Count, writable through the DBI
3 -	This field is reserved. Reserved
Extended_VC_Count	Extended VC Count The default value is the one less than the number of VCs that

48.8.45 Port VC Capability Register 2 (PCIE_RC_PVCCR2)

Offset: 0x140 + 0x8

Address: 1FF_C000h base + 148h offset = 1FF_C148h



PCIE_RC_PVCCR2 field descriptions

Field	Description
31–24 VC_Arbitration_Table_Offset	VC Arbitration Table Offset (not supported) The default value is 0x00 (no arbitration table present).
23–8 -	This field is reserved. Reserved
VC_Arbitration_Capability	VC Arbitration Capability Indicates which VC arbitration mode(s) the device supports, writable through the DBI: •Bit 0: Device supports hardware fixed arbitration scheme. For the core, the scheme is 16-phase weighted round robin (WRR). •Bit 1: Device supports 32-phase WRR

Table continues on the next page...

PCIE_RC_PVCCR2 field descriptions (continued)

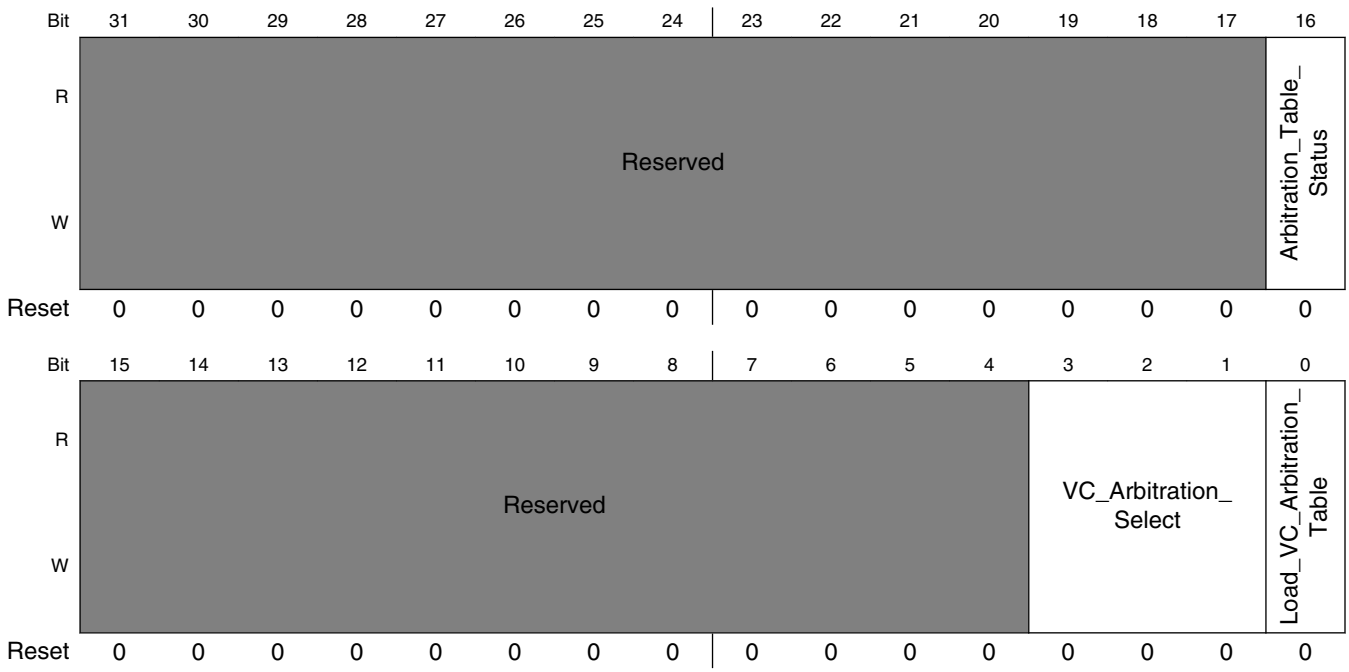
Field	Description
	•Bit 2: Device supports 64-phase WRR •Bit 3: Device supports 128-phase WRR •Bits 4-7: Reserved

48.8.46 Port VC Control and Status Register (PCIE_RC_PVCCSR)

Offset: 0x140 + 0xC

Bytes: 0-1

Address: 1FF_C000h base + 14Ch offset = 1FF_C14Ch



PCIE_RC_PVCCSR field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 Arbitration_Table_Status	Arbitration Table Status
15–4 -	This field is reserved. Reserved
3–1 VC_Arbitration_Select	VC Arbitration Select

Table continues on the next page...

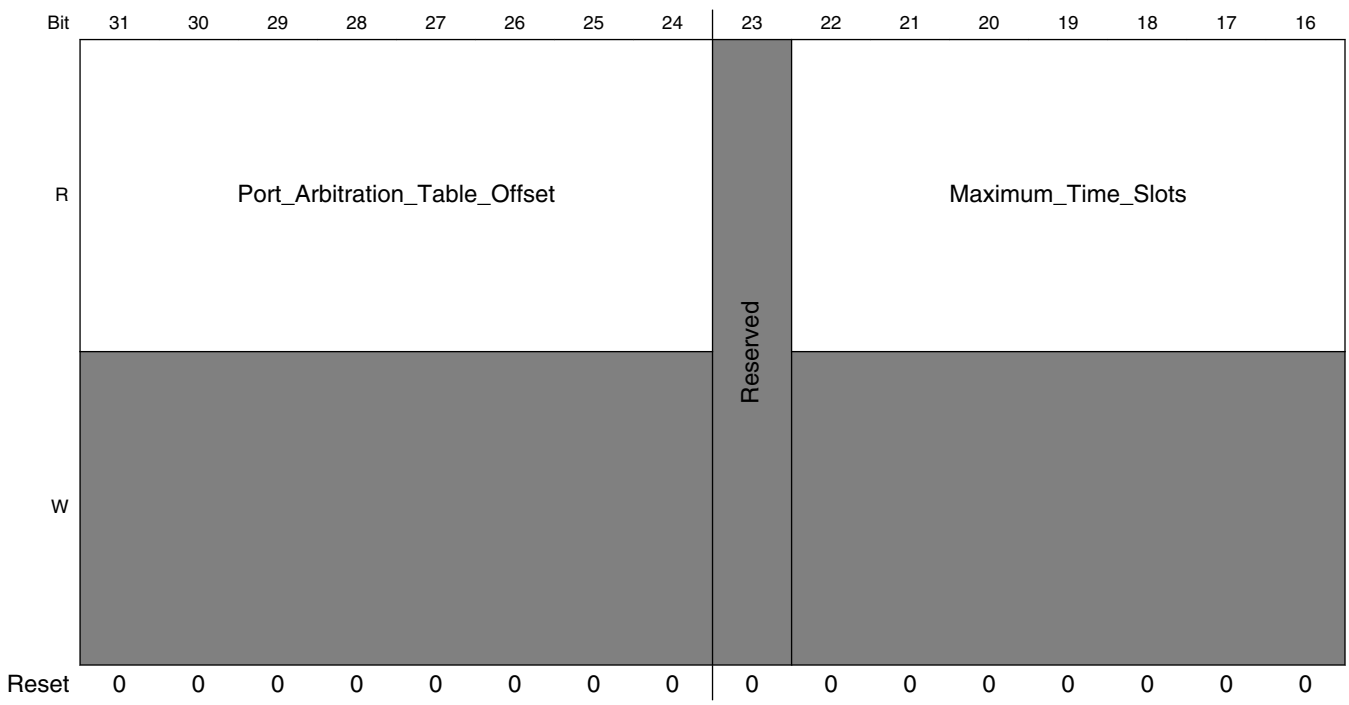
PCIE_RC_PVCCSR field descriptions (continued)

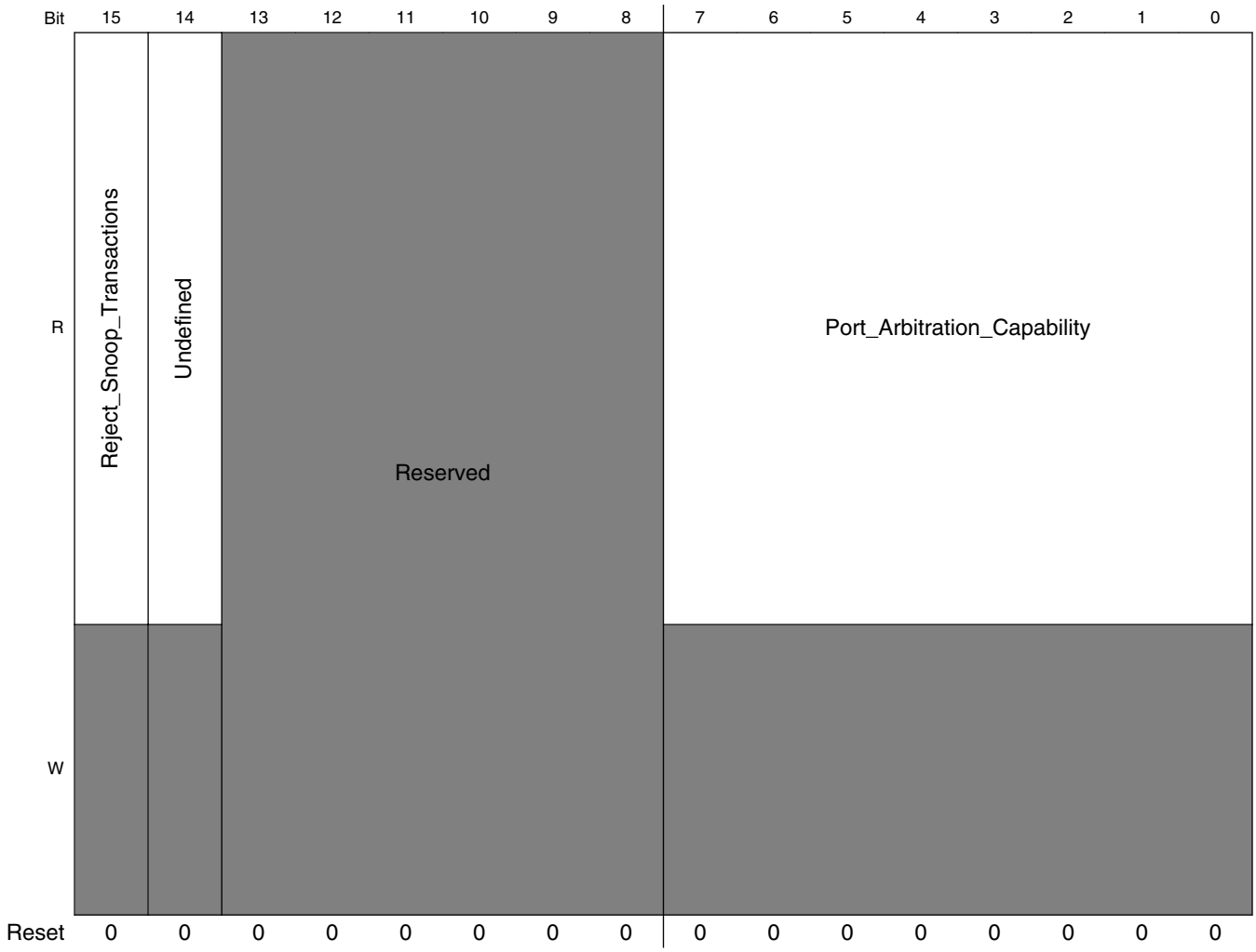
Field	Description
0 Load_VC_ Arbitration_Table	Load VC Arbitration Table

48.8.47 VC Resource Capability Register n (PCIE_RC_VCRCR)

Offset: 0x140 + 0x10

Address: 1FF_C000h base + 150h offset = 1FF_C150h





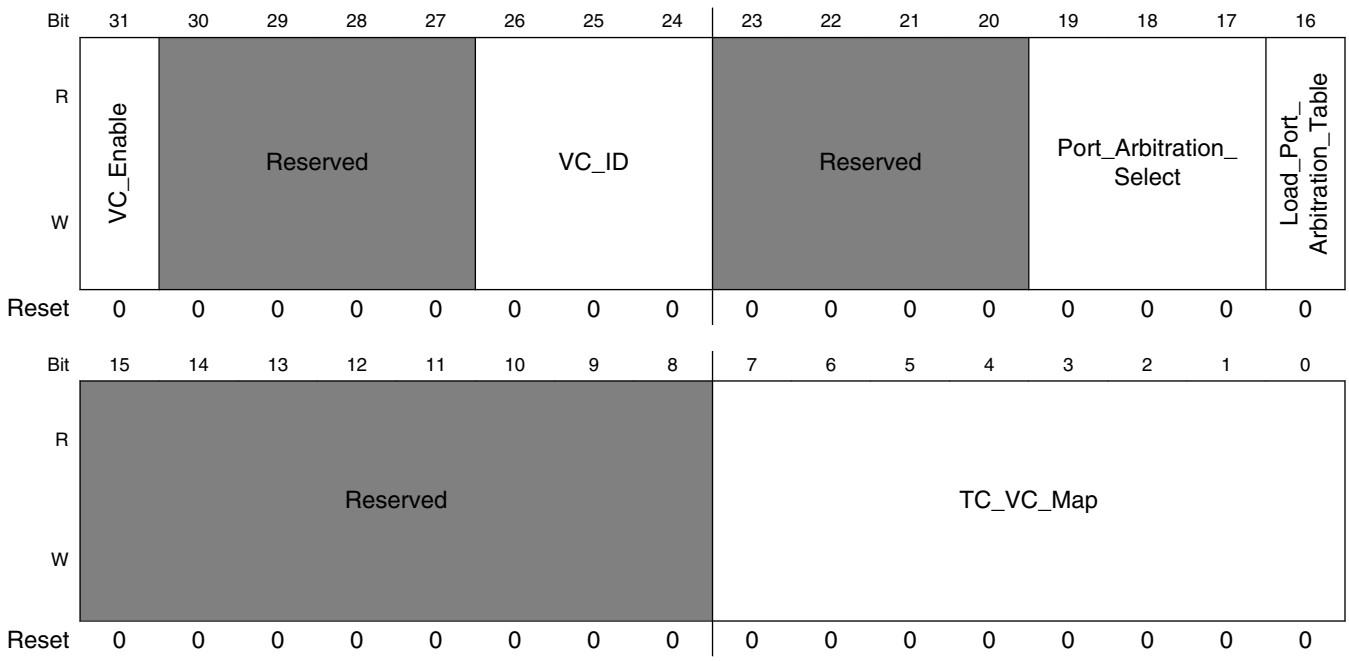
PCIE_RC_VCRCR field descriptions

Field	Description
31–24 Port_Arbitration_Table_Offset	Port Arbitration Table Offset
23 -	This field is reserved. Reserved
22–16 Maximum_Time_Slots	Maximum Time Slots
15 Reject_Snoop_Transactions	Reject Snoop Transactions
14 Undefined	Undefined for PCI Express 1.1 (Was Advanced Packet Switching for PCI Express 1.0a)
13–8 -	This field is reserved. Reserved
Port_Arbitration_Capability	Port Arbitration Capability

48.8.48 VC Resource Control Register n (PCIE_RC_VCRConR)

Offset: 0x140 + 0x14

Address: 1FF_C000h base + 154h offset = 1FF_C154h



PCIE_RC_VCRConR field descriptions

Field	Description
31 VC_Enable	VC Enable Hardwired to 1 for the first VC.
30–27 -	This field is reserved. Reserved
26–24 VC_ID	VC ID Hardwired to 0 for VC0.
23–20 -	This field is reserved. Reserved
19–17 Port_Arbitration_ Select	Port Arbitration Select
16 Load_Port_ Arbitration_Table	Load Port Arbitration Table
15–8 -	This field is reserved. Reserved
TC_VC_Map	TC/VC Map

Table continues on the next page...

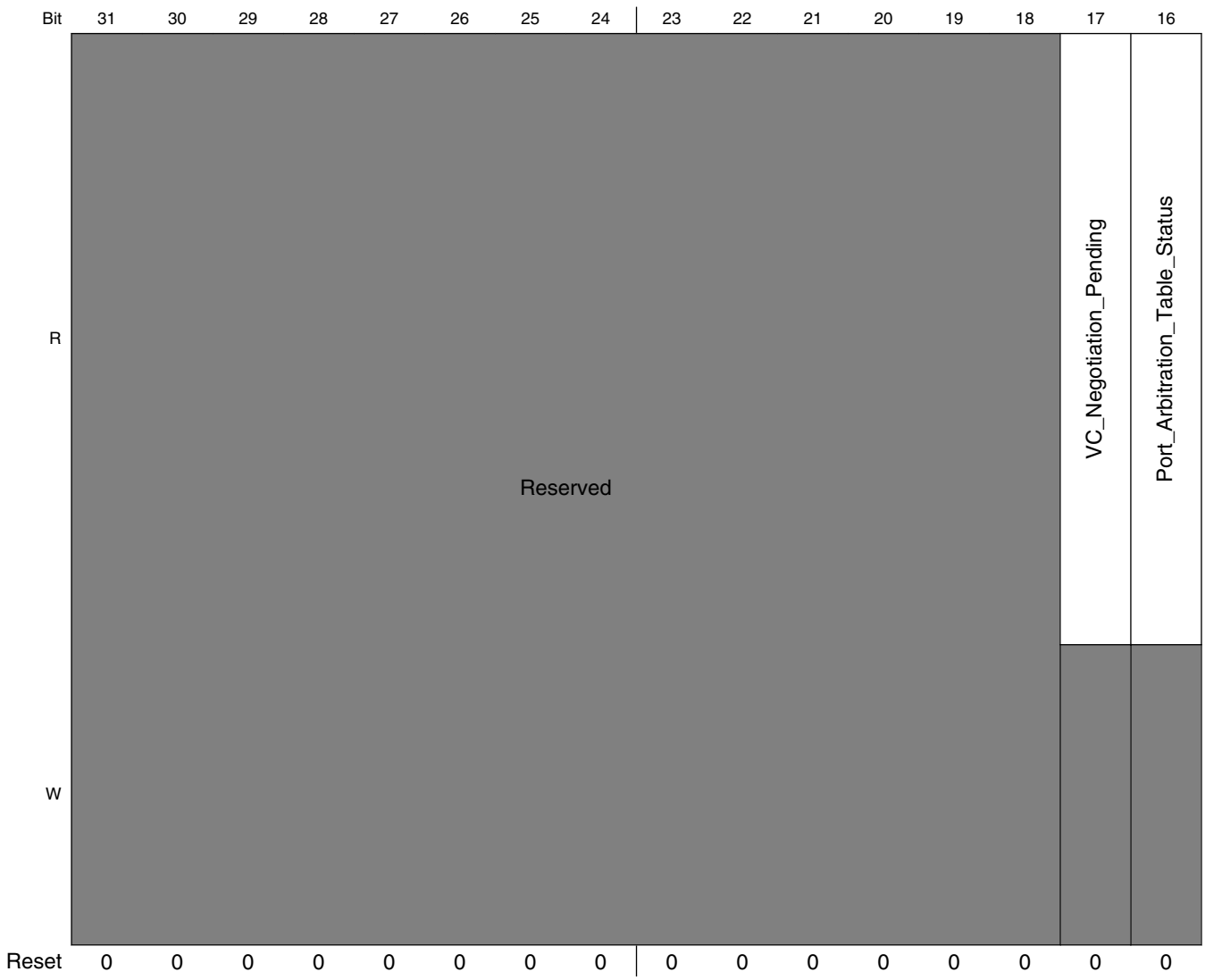
PCIE_RC_VCRConR field descriptions (continued)

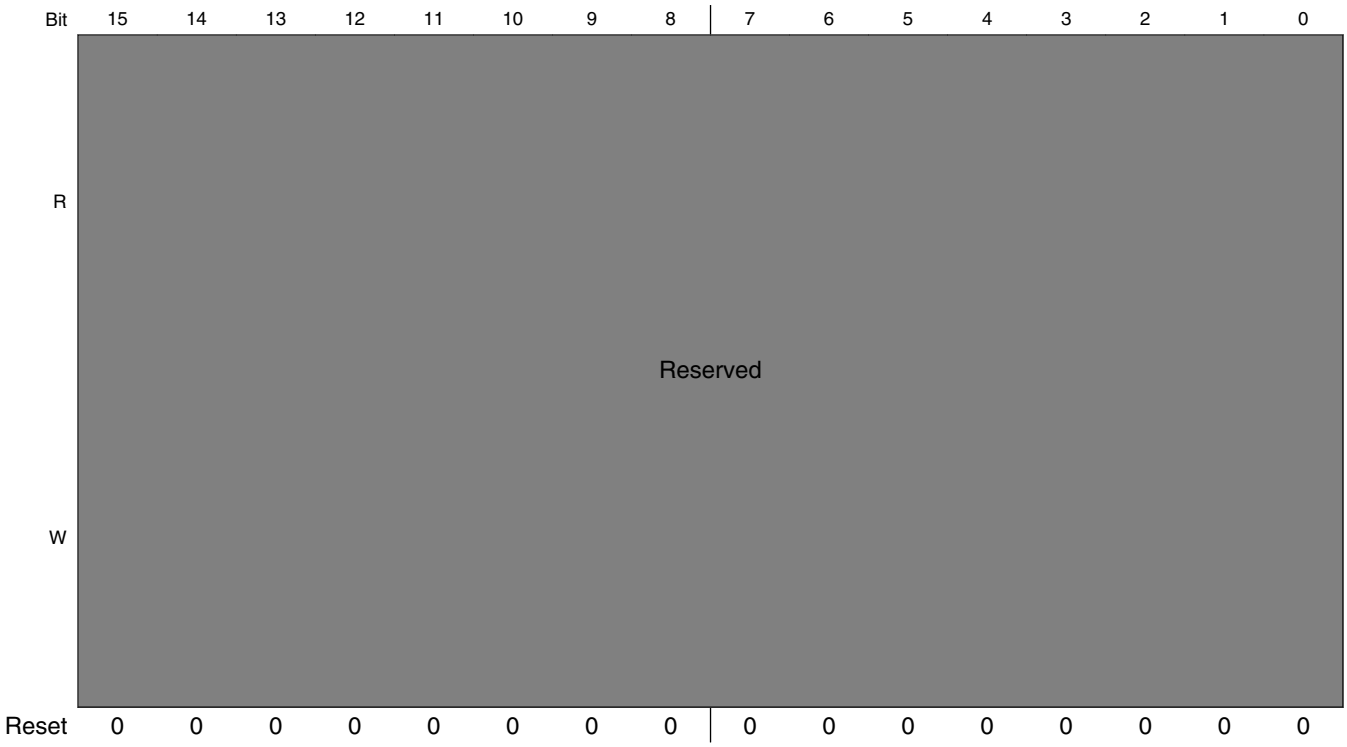
Field	Description
	Bit 0 is hardwired to 1; bits 7:1 are RW.

48.8.49 VC Resource Status Register n (PCIE_RC_VCRSR)

Offset: 0x140 + 0x18

Address: 1FF_C000h base + 158h offset = 1FF_C158h





PCIE_RC_VCRSR field descriptions

Field	Description
31–18 -	This field is reserved. Reserved
17 VC_Negotiation_Pending	VC Negotiation Pending
16 Port_Arbitration_Table_Status	Port Arbitration Table Status
-	This field is reserved. Reserved

48.9 PCIe CTRL Port Logic Memory Map/Register Definition

PCIE_PL memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1FF_C700	Ack Latency Timer and Replay Timer Register (PCIE_PL_ALTRTR)	32	R/W	03B4_3677h	48.9.1/4315
1FF_C704	Vendor Specific DLLP Register (PCIE_PL_VSDR)	32	R/W	FFFF_FFFFh	48.9.2/4316

Table continues on the next page...

PCIE_PL memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1FF_C708	Port Force Link Register (PCIE_PL_PFLR)	32	R/W	0700_0004h	48.9.3/4316
1FF_C70C	Ack Frequency and L0-L1 ASPM Control Register (PCIE_PL_AFLACR)	32	R/W	1B2C_2C00h	48.9.4/4318
1FF_C710	Port Link Control Register (PCIE_PL_PLCR)	32	R/W	0001_0020h	48.9.5/4320
1FF_C714	Lane Skew Register (PCIE_PL_LSR)	32	R/W	0000_0000h	48.9.6/4322
1FF_C718	Symbol Number Register (PCIE_PL_SNR)	32	R/W	0000_830Ah	48.9.7/4323
1FF_C71C	Symbol Timer Register and Filter Mask Register 1 (PCIE_PL_STRFM1)	32	R/W	0000_0640h	48.9.8/4324
1FF_C720	Filter Mask Register 2 (PCIE_PL_STRFM2)	32	R/W	0000_0000h	48.9.9/4328
1FF_C724	AMBA Multiple Outbound Decomposed NP Sub-Requests Control Register (PCIE_PL_AMODNPSR)	32	R/W	0000_0001h	48.9.10/ 4329
1FF_C728	Debug Register 0 (PCIE_PL_DEBUG0)	32	R	0000_0000h	48.9.11/ 4330
1FF_C72C	Debug Register 1 (PCIE_PL_DEBUG1)	32	R	0000_0000h	48.9.12/ 4330
1FF_C730	Transmit Posted FC Credit Status Register (PCIE_PL_TPFCSR)	32	R	0000_0000h	48.9.13/ 4331
1FF_C734	Transmit Non-Posted FC Credit Status Register (PCIE_PL_TNFCSR)	32	R	0000_0000h	48.9.14/ 4332
1FF_C738	Transmit Completion FC Credit Status Register (PCIE_PL_TCFCSR)	32	R	0000_0000h	48.9.15/ 4333
1FF_C73C	Queue Status Register (PCIE_PL_QSR)	32	R	0000_0000h	48.9.16/ 4334
1FF_C740	VC Transmit Arbitration Register 1 (PCIE_PL_VCTAR1)	32	R	0000_000Fh	48.9.17/ 4336
1FF_C744	VC Transmit Arbitration Register 2 (PCIE_PL_VCTAR2)	32	R	0000_0000h	48.9.18/ 4337
1FF_C748	VC0 Posted Receive Queue Control (PCIE_PL_VC0PRQC)	32	R/W	0010_C019h	48.9.19/ 4338
1FF_C74C	VC0 Non-Posted Receive Queue Control (PCIE_PL_VC0NRQC)	32	R/W	0020_0000h	48.9.20/ 4340
1FF_C750	VC0 Completion Receive Queue Control (PCIE_PL_VC0CRQC)	32	R/W	0080_0000h	48.9.21/ 4341
1FF_C754	VCn Posted Receive Queue Control (PCIE_PL_VCnPRQC)	32	R/W	0020_0000h	48.9.22/ 4342
1FF_C758	VCn Non-Posted Receive Queue Control (PCIE_PL_VCnNRQC)	32	R/W	0020_0000h	48.9.23/ 4344
1FF_C75C	VCn Completion Receive Queue Control (PCIE_PL_VCnCRQC)	32	R/W	0080_0000h	48.9.24/ 4345
1FF_C7A8	VC0 Posted Buffer Depth (PCIE_PL_VC0PBD)	32	R	000D_0065h	48.9.25/ 4346
1FF_C7AC	VC0 Non-Posted Buffer Depth (PCIE_PL_VC0NPBD)	32	R	000D_000Dh	48.9.26/ 4347

Table continues on the next page...

PCIE_PL memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1FF_C7B0	VC0 Completion Buffer Depth (PCIE_PL_VC0CBD)	32	R	0003_0003h	48.9.27/ 4348
1FF_C7B4	VCn Posted Buffer Depth (PCIE_PL_VC1PBD)	32	R	0000_0000h	48.9.28/ 4349
1FF_C7B8	VCn Non-Posted Buffer Depth (PCIE_PL_VC1NPBD)	32	R	0000_0000h	48.9.29/ 4350
1FF_C7BC	VCnCompletion Buffer Depth (PCIE_PL_VC1CBD)	32	R	0000_0000h	48.9.30/ 4351
1FF_C80C	Gen2 Control Register (PCIE_PL_G2CR)	32	R/W	0000_0001h	48.9.31/ 4351
1FF_C810	PHY Status (PCIE_PL_PHY_STATUS)	32	R	0000_0000h	48.9.32/ 4353
1FF_C814	PHY Control (PCIE_PL_PHY_CTRL)	32	R/W	0000_0000h	48.9.33/ 4353
1FF_C818	Master Response Composer Control Register 0 (PCIE_PL_MRCCR0)	32	R/W	0000_0302h	48.9.34/ 4354
1FF_C81C	Master Response Composer Control Register 1 (PCIE_PL_MRCCR1)	32	R/W	0000_0000h	48.9.35/ 4355
1FF_C820	MSI Controller Address (PCIE_PL_MSICA)	32	R/W	0000_0000h	48.9.36/ 4356
1FF_C824	MSI Controller Upper Address (PCIE_PL_MSICUA)	32	R/W	0000_0000h	48.9.37/ 4356
1FF_C828	MSI Controller Interrupt n Enable (PCIE_PL_MSICIn_ENB)	32	R/W	0000_0000h	48.9.38/ 4357
1FF_C82C	MSI Controller Interrupt n Mask (PCIE_PL_MSICIn_MASK)	32	R/W	0000_0000h	48.9.39/ 4357
1FF_C830	MSI Controller Interrupt nStatus (PCIE_PL_MSICIn_STATUS)	32	R/W	0000_0000h	48.9.40/ 4358
1FF_C888	MSI Controller General Purpose IO Register (PCIE_PL_MSICGPIO)	32	R/W	0000_0000h	48.9.41/ 4358
1FF_C900	iATU Viewport Register (PCIE_PL_iATUVR)	32	R/W	0000_0000h	48.9.42/ 4359
1FF_C904	iATU Region Control 1 Register (PCIE_PL_iATURC1)	32	R/W	0000_0000h	48.9.43/ 4360
1FF_C908	iATU Region Control 2 Register (PCIE_PL_iATURC2)	32	R/W	0000_0000h	48.9.44/ 4362
1FF_C90C	iATU Region Lower Base Address Register (PCIE_PL_iATURLBA)	32	R/W	0000_0000h	48.9.45/ 4365
1FF_C910	iATU Region Upper Base Address Register (PCIE_PL_iATURUBA)	32	R/W	0000_0000h	48.9.46/ 4366
1FF_C914	iATU Region Limit Address Register (PCIE_PL_iATURLA)	32	R/W	0000_FFFFh	48.9.47/ 4366
1FF_C918	iATU Region Lower Target Address Register (PCIE_PL_iATURLTA)	32	R/W	0000_0000h	48.9.48/ 4367

Table continues on the next page...

PCIE_PL memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1FF_C91C	iATU Region Upper Target Address Register (PCIE_PL_iATURUTA)	32	R/W	0000_0000h	48.9.49/ 4367

48.9.1 Ack Latency Timer and Replay Timer Register (PCIE_PL_ALTRTR)

Offset: 0x700

Address: 1FF_C000h base + 700h offset = 1FF_C700h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Replay_Time_Limit																Round_Trip_Latency_Time_Limit															
Reset	0	0	0	0	0	0	1	1	1	0	1	1	0	1	0	0	0	0	1	1	0	1	1	0	0	1	1	1	0	1	1	1

PCIE_PL_ALTRTR field descriptions

Field	Description
31–16 Replay_Time_ Limit	Replay Time Limit The replay timer expires when it reaches this limit. The core initiates a replay upon reception of a Nak or when the replay timer expires. The default value is 4143 / 2 = 2071.. The default is then updated based on the Negotiated Link Width and Max_Payload_Size. **NOTE:** If operating at 5 Gb/s, then an additional 51 / 2 = 26 is added. This is for additional internal processing for received TLPs and transmitted DLLPs.
Round_Trip_ Latency_Time_ Limit	Round Trip Latency Time Limit The Ack/Nak latency timer expires when it reaches this limit. The default value is 12429 / 2 = 5215. The default is then updated based on the Negotiated Link Width and Max_Payload_Size. **Note**: If operating at 5 Gb/s, then an additional 153 / 2 = 76 is added. This is for additional internal processing for received TLPs and transmitted DLLPs.

48.9.2 Vendor Specific DLLP Register (PCIE_PL_VSDR)

Offset: 0x700 + 0x4

Address: 1FF_C000h base + 704h offset = 1FF_C704h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Vendor_Specific_DLLP																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

PCIE_PL_VSDR field descriptions

Field	Description
Vendor_Specific_DLLP	Vendor Specific DLLP Register Used to send a specific PCI Express DLLP. The application writes the 8-bit DLLP Type and 24-bits of Payload data into this register, then sets bit 0 of Port Link Control Register (PCIE_PL_PLCR) to send the DLLP.

48.9.3 Port Force Link Register (PCIE_PL_PFLR)

Offset: 0x700 + 0x8

Address: 1FF_C000h base + 708h offset = 1FF_C708h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Low_Power_Entrance_Count								Reserved		Link_State					
Reset	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Force_Link	Reserved							Link_Number							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

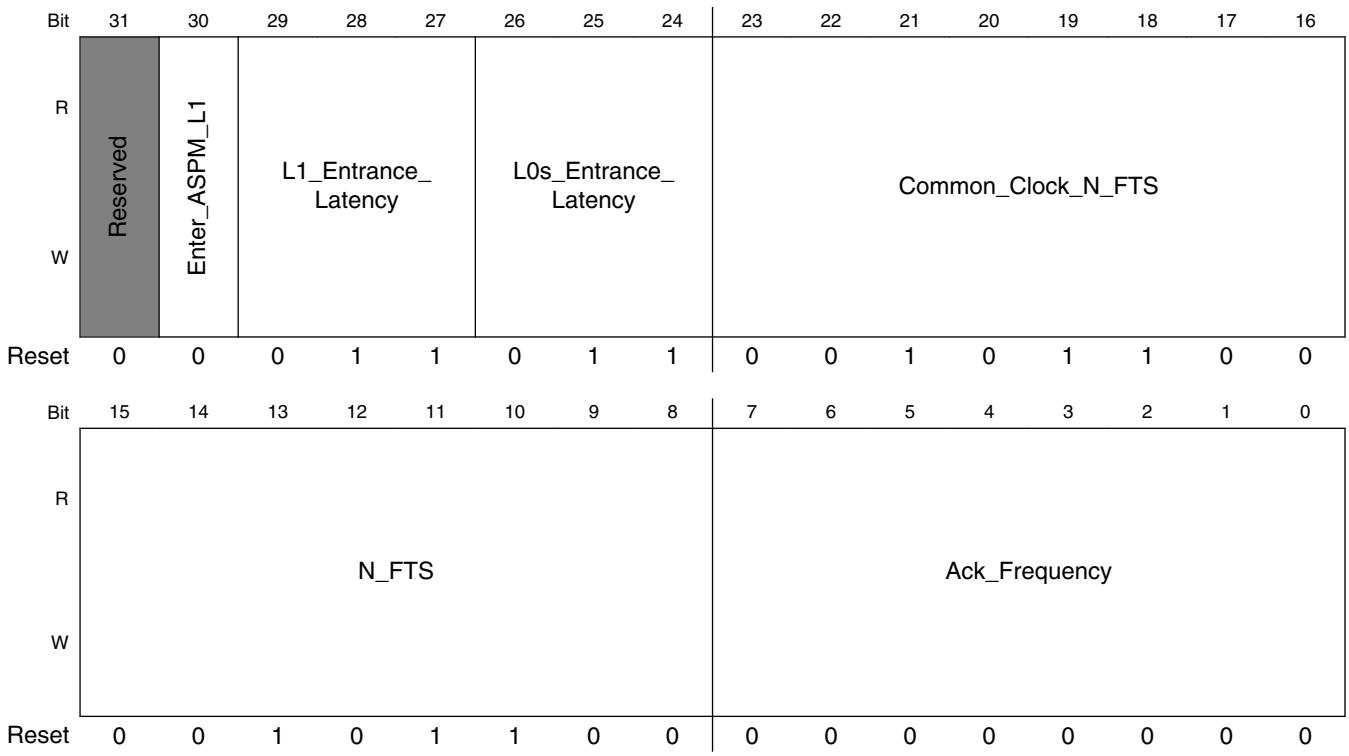
PCIE_PL_PFLR field descriptions

Field	Description
31–24 Low_Power_ Entrance_Count	Low Power Entrance Count The Power Management state will wait for this many clock cycles for the associated completion of a CfgWr to D-state register to go low-power. This register is intended for applications that do not let the core handle a completion for configuration request to the PMCSCR register. **Note**: Only used in the DM core (in EP mode), EP core, and the upstream port of a Switch.
23–22 -	This field is reserved. Reserved
21–16 Link_State	Link State The Link state that the core will be forced to when bit 15 (Force Link) is set. State encoding is defined in xmlh_ltssm.v.
15 Force_Link	Force Link Forces the Link to the state specified by the Link State field. The Force Link pulse will trigger Link re-negotiation. * Reading from this self-clearing register field always returns a 0.
14–8 -	This field is reserved. Reserved
Link_Number	Link Number Not used for Endpoint

48.9.4 Ack Frequency and L0-L1 ASPM Control Register (PCIE_PL_AFLACR)

Offset: 0x700 + 0xC

Address: 1FF_C000h base + 70Ch offset = 1FF_C70Ch



PCIE_PL_AFLACR field descriptions

Field	Description
31 -	This field is reserved. Reserved
30 Enter_ASPM_L1	Enter ASPM L1 without receive in L0s. Allow core to enter ASPM L1 even when link partner did not go to L0s (receive is not in L0s). When not set, core goes to ASPM L1 only after idle period during which both receive and transmit are in L0s.
29–27 L1_Entrance_ Latency	L1 Entrance Latency Values correspond to: 000 1 ìs 001 2 ìs 010 4 ìs 011 8 ìs

Table continues on the next page...

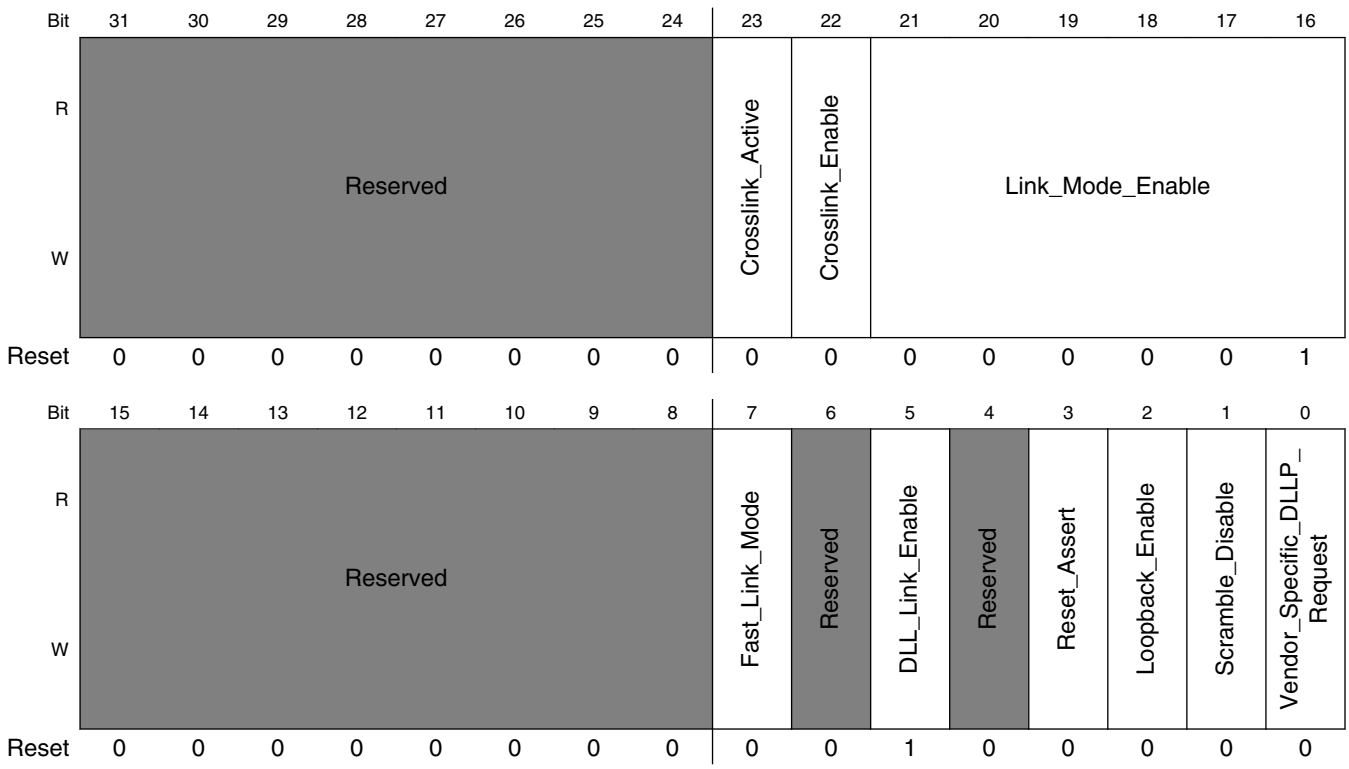
PCIE_PL_AFLACR field descriptions (continued)

Field	Description
	100 16 ìs 101 32 ìs 110 64 ìs 111 64 ìs
26–24 L0s_Entrance_ Latency	L0s Entrance Latency Values correspond to: 000 1 ìs 001 2 ìs 010 3 ìs 011 4 ìs 100 5 ìs 101 6 ìs 110 7 ìs 111 7 ìs
23–16 Common_Clock_ N_FTS	Common Clock N_FTS This is the N_FTS when common clock is used. The number of Fast Training Sequence ordered sets to be transmitted when transitioning from L0s to L0. The maximum number of FTS ordered-sets that a component can request is 255. This field is writable only if the parameters are selected as follows during configuration of the core; CX_NFTS != CX_COMM_NFTS DEFAULT_L0S_EXIT_LATENCY != DEFAULT_COMM_L0S_EXIT_LATENCY DEFAULT_L1_EXIT_LATENCY != DEFAULT_COMM_L1_EXIT_LATENCY otherwise, it will be hard coded to the value of the CX_COMM_NFTS configuration parameter. **Note**: The core does not support a value of zero; a value of zero can cause the LTSSM to go into the recovery state when exiting from L0s.
15–8 N_FTS	N_FTS The number of Fast Training Sequence ordered sets to be transmitted when transitioning from L0s to L0. The maximum number of FTS ordered-sets that a component can request is 255. **Note**: The core does not support a value of zero; a value of zero can cause the LTSSM to go into the recovery state when exiting from L0s.
Ack_Frequency	Ack Frequency The core accumulates the number of pending Ack's specified here (up to 255) before sending an Ack DLLP see Link Layer: (Flow Control and ACK/NAK DLLPs) for more details.

48.9.5 Port Link Control Register (PCIE_PL_PLCR)

Offset: 0x700 + 0x10

Address: 1FF_C000h base + 710h offset = 1FF_C710h



PCIE_PL_PLCR field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23 Crosslink_Active	Crosslink Active. Indicates a change from upstream to downstream or downstream to upstream. Same as the xmlh_crosslink_active output.
22 Crosslink_Enable	Crosslink Enable
21–16 Link_Mode_ Enable	Link Mode Enable The default value is the number of Lanes supported in the version of the core you are using. 000001 x1 000011 x2 000111 x4 001111 x8

Table continues on the next page...

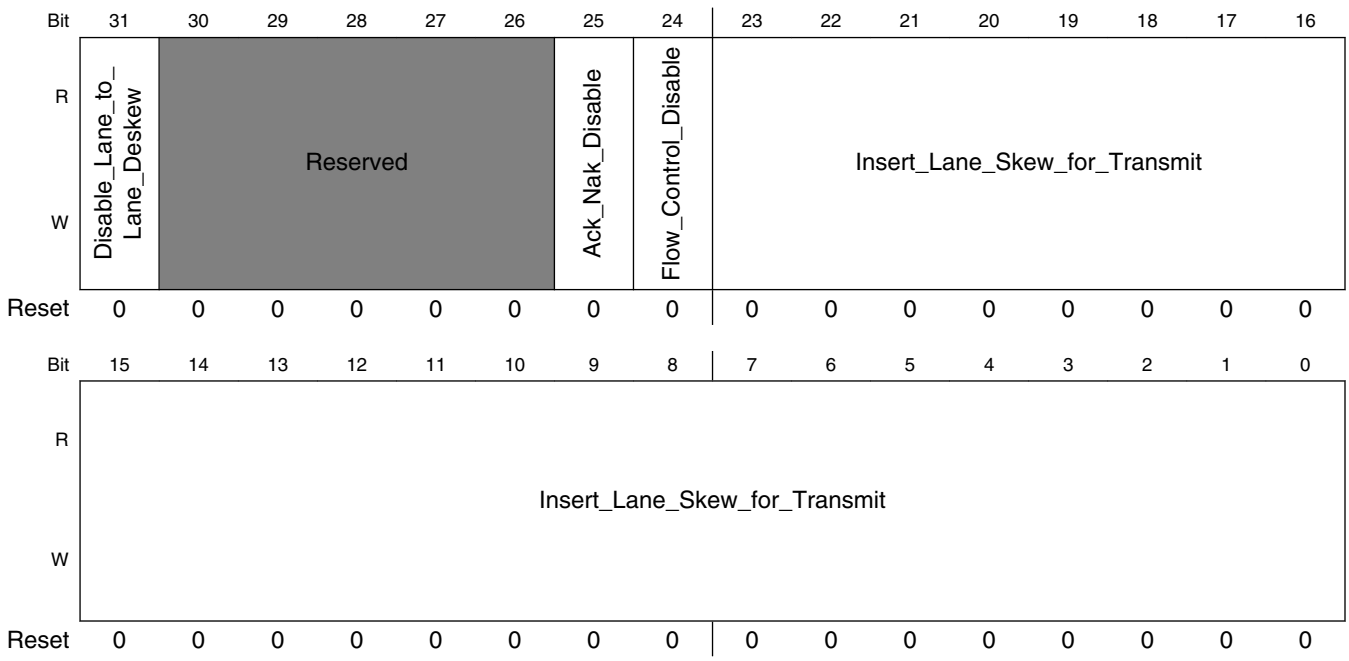
PCIE_PL_PLCR field descriptions (continued)

Field	Description
	011111 x16 111111 x32 (not supported)
15–8 -	This field is reserved. Reserved
7 Fast_Link_Mode	Fast Link Mode Sets all internal timers to Fast Mode for speeding up simulation. Forces the LTSSM training (link initialization) to use shorter time-outs and to link up faster. The scaling factor is 1024 for all internal timers. **Note**: Fast Link Mode can also be activated by setting the diag_ctrl_bus[2] pin to '1'.
6 -	This field is reserved. Reserved
5 DLL_Link_Enable	DLL Link Enable Enables Link initialization. If DLL Link Enable = 0, the core does not transmit InitFC DLLPs and does not establish a Link.
4 -	This field is reserved. Reserved
3 Reset_Assert	Reset Assert Triggers a recovery and forces the LTSSM to the Hot Reset state (downstream port only).
2 Loopback_ Enable	Loopback Enable Turns on loopback.
1 Scramble_ Disable	Scramble Disable Turns off data scrambling.
0 Vendor_Specific_ DLLP_Request	Vendor Specific DLLP Request When software writes a '1' to this bit, the core transmits the DLLP contained in the Vendor Specific DLLP Register (PCIE_PL_VSDR). * Reading from this self-clearing register field always returns a 0.

48.9.6 Lane Skew Register (PCIE_PL_LSR)

Offset: 0x700 + 0x14

Address: 1FF_C000h base + 714h offset = 1FF_C714h



PCIE_PL_LSR field descriptions

Field	Description
31 Disable_Lane_to_Lane_Deskew	Disable Lane-to-Lane Deskew Causes the core to disable the internal Lane-to-Lane deskew logic.
30–26 -	This field is reserved. Reserved
25 Ack_Nak_Disable	Ack/Nak Disable Prevents the core from sending Ack and Nak DLLPs.
24 Flow_Control_Disable	Flow Control Disable Prevents the core from sending FC DLLPs.
Insert_Lane_Skew_for_Transmit	Insert Lane Skew for Transmit (not supported for x16) Optional feature that causes the core to insert skew between Lanes for test purposes. There are three bits per Lane. The value is in units of one symbol time. For example, the value 010b for a Lane forces a skew of two symbol times for that Lane. The maximum skew value for any Lane is 5 symbol times.

48.9.7 Symbol Number Register (PCIE_PL_SNR)

Offset: 0x700 + 0x18

Address: 1FF_C000h base + 718h offset = 1FF_C718h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Configuration_ Requests			Timer_Modifier_for_Flow_Control_ Watchdog_Timer					Timer_Modifier_for_Ack_Nak_ Latency_Timer					Timer_Modifier_for_ Replay_Timer		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Timer_ Modifier_for_ Replay_ Timer		Reserved			Number_of_SKP_ Symbols			Reserved				Number_of_TS_Symbols			
Reset	1	0	0	0	0	0	1	1	0	0	0	0	1	0	1	0

PCIE_PL_SNR field descriptions

Field	Description
31–29 Configuration_ Requests	Configuration Requests targeted at function numbers above this value will be returned with UR (unsupported request).
28–24 Timer_Modifier_ for_Flow_ Control_ Watchdog_Timer	Timer Modifier for Flow Control Watchdog Timer Increases the timer value for the Flow Control watchdog timer, in increments of 16 clock cycles.
23–19 Timer_Modifier_ for_Ack_Nak_ Latency_Timer	Timer Modifier for Ack/Nak Latency Timer Increases the timer value for the Ack/Nak latency timer, in increments of 64 clock cycles.
18–14 Timer_Modifier_ for_Replay_Timer	Timer Modifier for Replay Timer Increases the timer value for the replay timer, in increments of 64 clock cycles.
13–11 -	This field is reserved. Reserved
10–8 Number_of_SKP_ Symbols	Number of SKP Symbols
7–4 -	This field is reserved. Reserved
Number_of_TS_ Symbols	Number of TS Symbols Sets the number of TS identifier symbols that are sent in TS1 and TS2 ordered sets.

48.9.8 Symbol Timer Register and Filter Mask Register 1 (PCIE_PL_STRFM1)

Offset: 0x700 + 0x1C

Table 48-127. Filter Mask 1: Mask RADM Filtering and Error Handling Rules

Bits	Name	Name
31	CX_FLT_MASK_RC_CFG_DISCARD	CX_FLT_MASK_RC_CFG_DISCARD 0 For RADM RC filter to not allow CFG transaction being received 1 For RADM RC filter to allow CFG transaction being received
30	CX_FLT_MASK_RC_IO_DISCARD	CX_FLT_MASK_RC_IO_DISCARD 0 For RADM RC filter to not allow IO transaction being received 1 For RADM RC filter to allow IO transaction being received
29	CX_FLT_MASK_MSG_DROP	CX_FLT_MASK_MSG_DROP 0 Drop MSG TLP (except for Vendor MSG) 1 Do not Drop MSG (except for Vendor MSG)
28	CX_FLT_MASK_CPL_ECRC_DISCARD	CX_FLT_MASK_CPL_ECRC_DISCARD 0 Discard TLPs with ECRC errors for CPL type 1 Allow TLPs with ECRC errors to be passed up for CPL type
27	CX_FLT_MASK_ECRC_DISCARD	CX_FLT_MASK_ECRC_DISCARD 0 Discard TLPs with ECRC errors 1 Allow TLPs with ECRC errors to be passed up
26	CX_FLT_MASK_CPL_LEN_MATCH	CX_FLT_MASK_CPL_LEN_MATCH 0 Enforce length match for received CPL TLPs; a violation results in cpl_abort, and possibly AER of unexp_cpl_err 1 MASK length match for received CPL TLPs
25	CX_FLT_MASK_CPL_ATTR_MATCH	CX_FLT_MASK_CPL_ATTR_MATCH

Table continues on the next page...

Table 48-127. Filter Mask 1: Mask RADM Filtering and Error Handling Rules (continued)

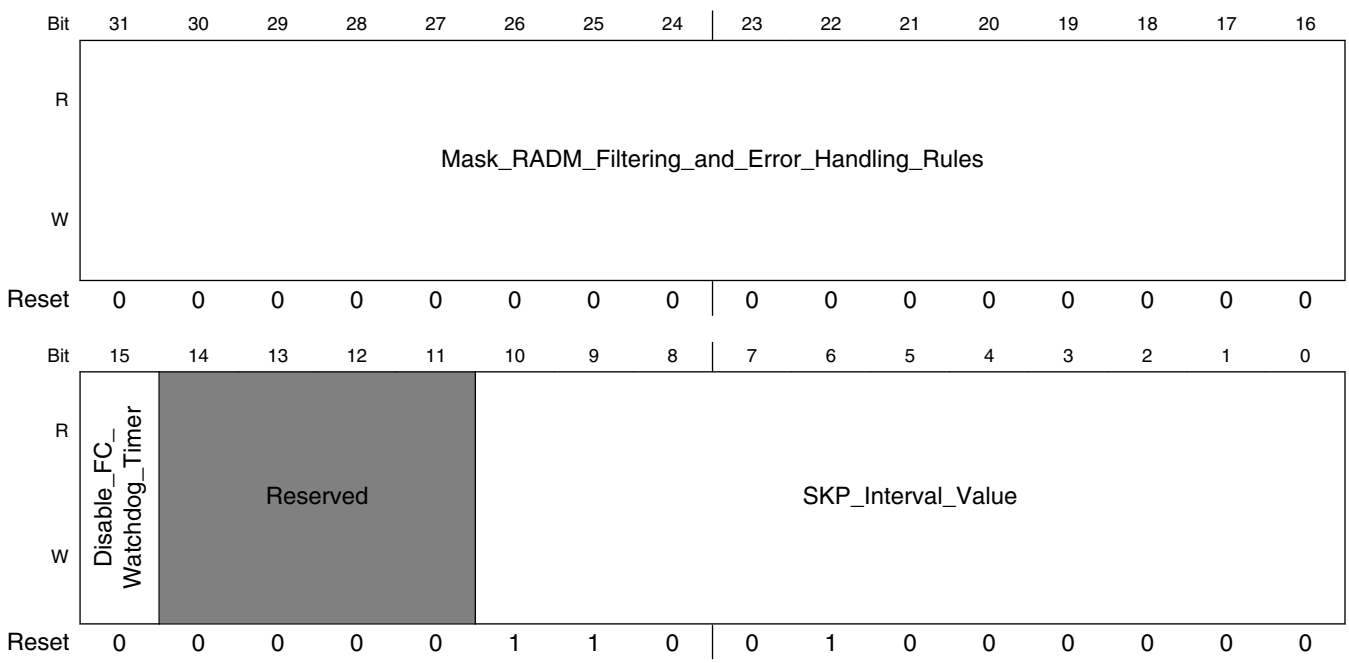
Bits	Name	Name
		0 Enforce attribute match for received CPL TLPs; a violation results in a malformed TLP error, and possibly AER of unexp_cpl_err, cpl_rcvd_ur, cpl_rcvd_ca 1 Mask attribute match for received CPL TLPs
24	CX_FLT_MASK_CPL_TC_MATCH	CX_FLT_MASK_CPL_TC_MATCH 0 Enforce Traffic Class match for received CPL TLPs; a violation results in a malformed TLP error, and possibly AER of unexp_cpl_err, cpl_rcvd_ur, cpl_rcvd_ca 1 Mask Traffic Class match for received CPL TLPs
23	CX_FLT_MASK_CPL_FUNC_MATCH	CX_FLT_MASK_CPL_FUNC_MATCH 0 Enforce function match for received CPL TLPs; a violation results in cpl_abort, and possibly AER of unexp_cpl_err, cpl_rcvd_ur, cpl_rcvd_ca 1 Mask function match for received CPL TLPs
22	CX_FLT_MASK_CPL_REQID_MATCH	CX_FLT_MASK_CPL_REQID_MATCH 0 Enforce Req. Id match for received CPL TLPs; a violation result in cpl_abort, and possibly AER of unexp_cpl_err, cpl_rcvd_ur, cpl_rcvd_ca 1 Mask Req. Id match for received CPL TLPs
21	CX_FLT_MASK_CPL_TAGERR_MATCH	CX_FLT_MASK_CPL_TAGERR_MATCH 0 Enforce Tag Error Rules for received CPL TLPs; a violation result in cpl_abort, and possibly AER of unexp_cpl_err, cpl_rcvd_ur, cpl_rcvd_ca 1 Mask Tag Error Rules for received CPL TLPs
20	CX_FLT_MASK_LOCKED_RD_AS_UR	CX_FLT_MASK_LOCKED_RD_AS_UR 0 Treat locked Read TLPs as UR for EP; Supported for RC 1 Treat locked Read TLPs as Supported for EP; UR for RC

Table continues on the next page...

Table 48-127. Filter Mask 1: Mask RADM Filtering and Error Handling Rules (continued)

Bits	Name	Name
19	CX_FLT_MASK_CFG_TYPE1_RE_AS_UR	CX_FLT_MASK_CFG_TYPE1_RE_AS_UR 0 Treat CFG type1 TLPs as UR for EP; Supported for RC 1 Treat CFG type1 TLPs as Supported for EP; UR for RC If CX_SRIOV_ENABLE is set then this bit is set to allow the filter to process Type 1 Config requests if the EP consumes more than one bus number.
18	CX_FLT_MASK_UR_OUTSIDE_BAR	CX_FLT_MASK_UR_OUTSIDE_BAR 0 Treat out-of-bar TLPs as UR 1 Treat out-of-bar TLPs as Supported Requests
17	CX_FLT_MASK_UR_POIS	CX_FLT_MASK_UR_POIS 0 Treat poisoned TLPs as UR 1 Treat poisoned TLPs as Supported Requests
16	CX_FLT_MASK_UR_FUNC_MISMATCH	CX_FLT_MASK_UR_FUNC_MISMATCH 0 Treat Function MisMatched TLPs as UR 1 Treat Function MisMatched TLPs as Supported

Address: 1FF_C000h base + 71Ch offset = 1FF_C71Ch



PCIE_PL_STRFM1 field descriptions

Field	Description
31–16 Mask_RADM_Filtering_and_Error_Handling_Rules	Mask RADM Filtering and Error Handling Rules: Mask 1 There are several mask bits to turn off the filtering and error handling rules In each case, 0 applies the associated filtering rule and 1 masks the associated filtering rule. A more detailed description for these bits is provided in Table 5-348. [31]: Mask filtering of received Configuration Requests (RC mode only) [30]: Mask filtering of received I/O Requests (RC mode only) [29]: Send Message TLPs to the application on RTRGT1 and send decoded Message on the SII (1) or send decoded Message on the SII, then drop the Message TLPs (0). The default value for this bit is the inverse of `FLT_DROP_MSG. That is, if `FLT_DROP_MSG = 1, then the default value of this bit is 0 (drop Message TLPs). Note that this bit only controls message TLPs other than Vendor MSGs. Vendor MSGs are controlled by Filter Mask Register 2 (PCIE_PL_STRFM2), bits [1:0]. [28]: Mask ECRC error filtering for Completions [27]: Mask ECRC error filtering [26]: Mask Length mismatch error for received Completions [25]: Mask Attributes mismatch error for received Completions [24]: Mask Traffic Class mismatch error for received Completions [23]: Mask function mismatch error for received Completions [23]: Mask Requester ID mismatch error for received Completions [21]: Mask Tag error rules for received Completions [20]: Mask Locked Request filtering [19]: Mask Type 1 Configuration Request filtering [18]: Mask BAR match filtering [17]: Mask poisoned TLP filtering [16]: Mask function mismatch filtering for incoming Requests
15 Disable_FC_Watchdog_Timer	Disable FC Watchdog Timer
14–11 -	This field is reserved. Reserved
SKP_Interval_Value	SKP Interval Value The number of symbol times to wait between transmitting SKP ordered sets. Note that the core actually waits the number of symbol times in this register plus 1 between transmitting SKP ordered sets. The application must program this register accordingly. For example, if 1536 we're programmed into this register (in a 250MHz core), then the core will actually transmit Skp ordered sets once every 1537 symbol times. Also, the value programmed to this register is actually clock ticks and not symbol times. In a 125MHz core, programming the value programmed to this register should be scaled down by a factor of 2 (since 1 clock tick=2 symbol times in this case).

48.9.9 Filter Mask Register 2 (PCIE_PL_STRFM2)

Offset: 0x700 + 0x20

Address: 1FF_C000h base + 720h offset = 1FF_C720h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Mask_RADM_Filtering_and_Error_Handling_Rules																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

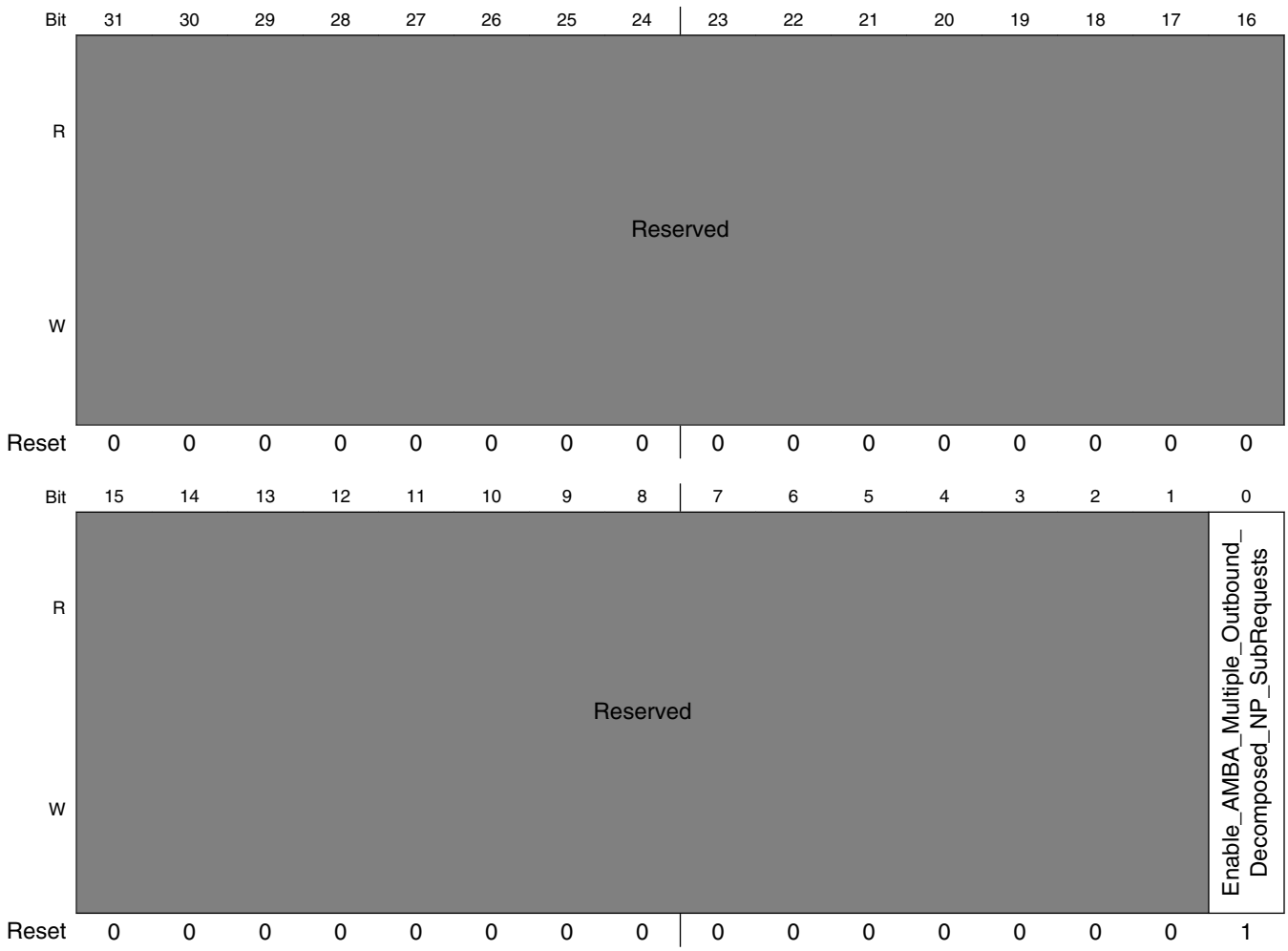
PCIE_PL_STRFM2 field descriptions

Field	Description
Mask_RADM_ Filtering_and_ Error_Handling_ Rules	Mask RADM Filtering and Error Handling Rules: Mask 2 It defaults to 0x0 There are several mask bits used to turn off the filtering and error handling rules . •[31:4]: Reserved •[3]: `CX_FLT_MASK_HANDLE_FLUSH - 0: Disable Core Filter to handle flush request - 1: Enable Core Filter to handle flush request •[2]: `CX_FLT_MASK_DABORT_4UCPL - 0: Enable DLLP abort for unexpected CPL - 1: Do not enable DLLP abort for unexpected CPL •[1]: `CX_FLT_MASK_VENMSG1_DROP - 0: Vendor MSG Type 1 dropped silently - 1: Vendor MSG Type 1 not dropped •[0]: `CX_FLT_MASK_VENMSG0_DROP - 0: Vendor MSG Type 0 dropped with UR error reporting - 1: Vendor MSG Type 0 not dropped

48.9.10 AMBA Multiple Outbound Decomposed NP Sub-Requests Control Register (PCIE_PL_AMODNPSR)

Offset: 0x700 + 0x24

Address: 1FF_C000h base + 724h offset = 1FF_C724h



PCIE_PL_AMODNPSR field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 Enable_AMBA_Multiple_Outbound_Decomposed_	Enable AMBA Multiple Outbound Decomposed NP Sub- Requests. This bit when set to '0' disables the possibility of having multiple outstanding non-posted requests that were derived from decomposition of an outbound AMBA request. See Supported AXI Burst Operations for more details.

Table continues on the next page...

PCIE_PL_AMODNPSR field descriptions (continued)

Field	Description
NP_ SubRequests	You should not clear this register unless your application master is requesting an amount of read data greater than Max_Read_Request_Size, and the remote device (or switch) is reordering completions that have different tags

48.9.11 Debug Register 0 (PCIE_PL_DEBUG0)

Offset: 0x700 + 0x28

[31:28]: rmlh_ts_link_ctrl Link control bits advertised by link partner

[27]: rmlh_ts_lane_num_is_k23 Currently receiving k237 (PAD) in place of lane number

[26]: rmlh_ts_link_num_is_k23 Currently receiving k237 (PAD) in place of link number

[25]: rmlh_rcvd_idle[0] Receiver is receiving logical idle

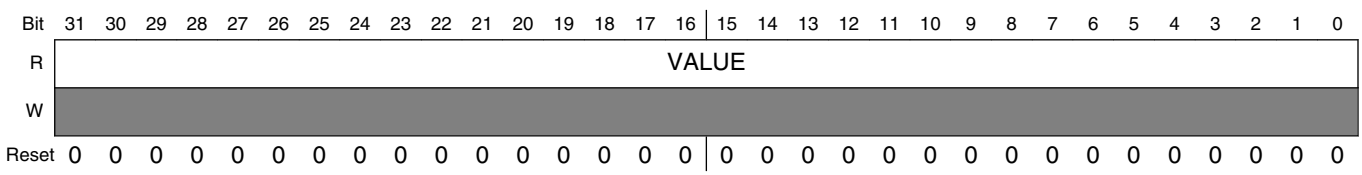
[24]: rmlh_rcvd_idle[1] 2nd symbol is also idle (16bit PHY interface only)

[23:8]: mac_phy_txdata PIPE Transmit data

[7:6]: mac_phy_txdatak PIPE transmit K indication

[5:0]: xmlh_ltssm_state LTSSM current state. See source for encodings

Address: 1FF_C000h base + 728h offset = 1FF_C728h



PCIE_PL_DEBUG0 field descriptions

Field	Description
VALUE	The value on cxpl_debug_info[31:0].

48.9.12 Debug Register 1 (PCIE_PL_DEBUG1)

Offset: 0x700 + 0x2C

[63]: xmlh_scrambler_disable Scrambling disabled for the link

[62]: xmlh_link_disable LTSSM in DISABLE state. Link inoperable

[61]: xmlh_link_in_training LTSSM performing link training

[60]: xmlh_rcvr_revrs_pol_en LTSSM testing for polarity reversal

[59]: xmlh_training_rst_n LTSSM-negotiated link reset

[58:55]: 0000b Constant/reserved

[54]: mac_phy_txdetectrx_loop PIPE receiver detect/loopback request

[53]: mac_phy_txelecidle[0] PIPE transmit electrical idle request

[52]: mac_phy_txcompliance[0] PIPE transmit compliance request

[51]: app_init_rst Application request to initiate training reset

[50:48]: 000b Constant/reserved

[47:40]: rmlh_ts_link_num Link number advertised/confirmed by link partner

[39:38]: 00b Constant/reserved

[37]: xmtbyte_skip_sent A skip ordered set has been transmitted

[36]: xmlh_link_up LTSSM reports PHY link up

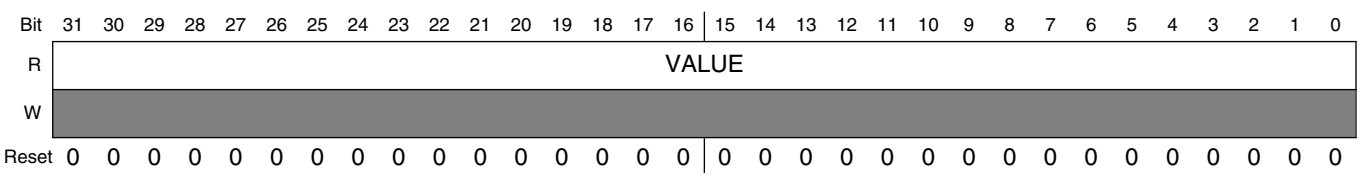
[35]: rmlh_inskip_rcv Receiver reports skip reception

[34]: rmlh_ts1_rcvd TS1 training sequence received (pulse)

[33]: rmlh_ts2_rcvd TS2 training sequence received (pulse)

[32]: rmlh_rcvd_lane_rev Receiver detected lane reversal

Address: 1FF_C000h base + 72Ch offset = 1FF_C72Ch



PCIE_PL_DEBUG1 field descriptions

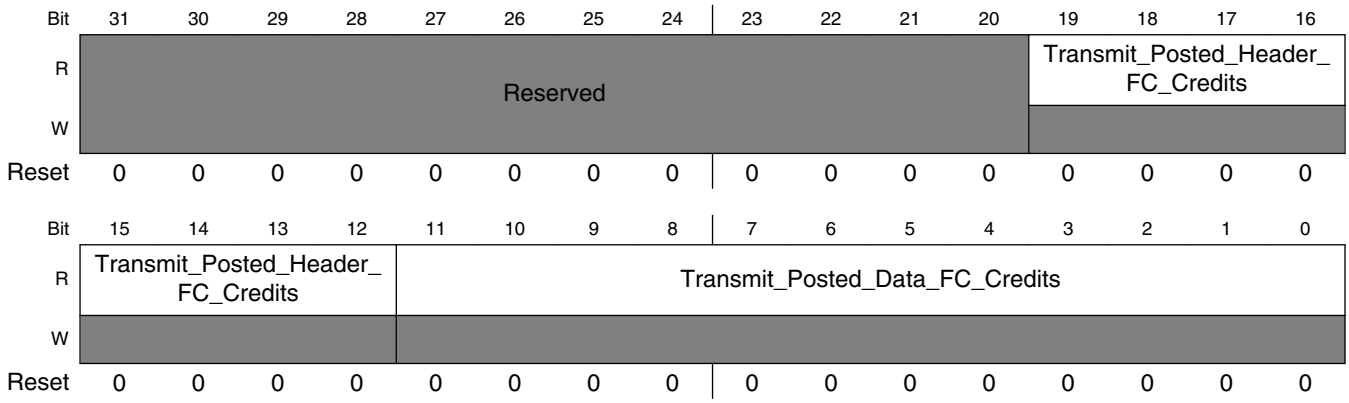
Field	Description
VALUE	The value on cxpl_debug_info[63:32].

48.9.13 Transmit Posted FC Credit Status Register (PCIE_PL_TPFCSR)

Offset: 0x700 + 0x30

*Default value depends on the number of advertised credits for header and data {12'b0, xtlh_xadm_ph_cdts, xtlh_xadm_pd_cdts}; If the number of advertised completion credits (both header and data) are infinite, then the default would be {12'b0, 8'hFF, 12'hFFF}.

Address: 1FF_C000h base + 730h offset = 1FF_C730h



PCIE_PL_TPFCSR field descriptions

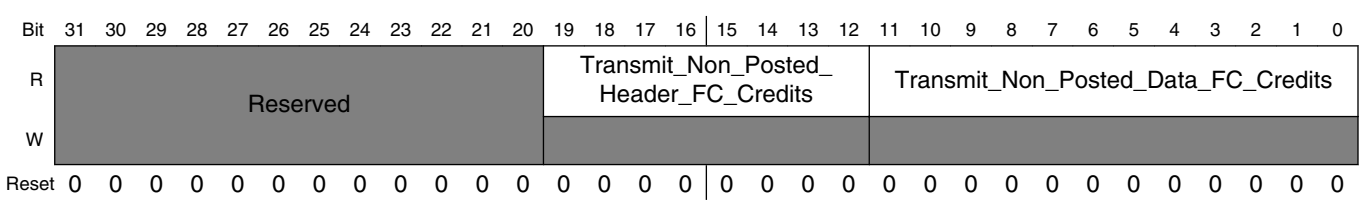
Field	Description
31–20 -	This field is reserved. Reserved
19–12 Transmit_ Posted_Header_ FC_Credits	Transmit Posted Header FC Credits The Posted Header credits advertised by the receiver at the other end of the Link, updated with each UpdateFC DLLP.
Transmit_ Posted_Data_ FC_Credits	Transmit Posted Data FC Credits The Posted Data credits advertised by the receiver at the other end of the Link, updated with each UpdateFC DLLP.

48.9.14 Transmit Non-Posted FC Credit Status Register (PCIE_PL_TNFCSR)

Offset: 0x700 + 0x34

*Default value depends on the number of advertised credits for header and data {12'b0, xtlh_xadm_nph_cdts, xtlh_xadm_npd_cdts}; If the number of advertised completion credits (both header and data) are infinite, then the default would be {12'b0, 8'hFF, 12'hFFF}.

Address: 1FF_C000h base + 734h offset = 1FF_C734h



PCIE_PL_TNFCSR field descriptions

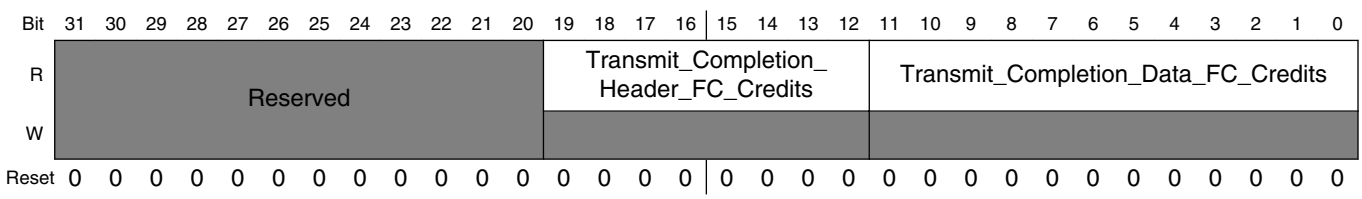
Field	Description
31–20 -	This field is reserved. Reserved
19–12 Transmit_Non_Posted_Header_FC_Credits	Transmit Non-Posted Header FC Credits The Non-Posted Header credits advertised by the receiver at the other end of the Link, updated with each UpdateFC DLLP.
Transmit_Non_Posted_Data_FC_Credits	Transmit Non-Posted Data FC Credits The Non-Posted Data credits advertised by the receiver at the other end of the Link, updated with each UpdateFC DLLP.

48.9.15 Transmit Completion FC Credit Status Register (PCIE_PL_TCFCSR)

Offset: 0x700 + 0x38

*Default value depends on the number of advertised credits for header and data {12'b0, xtlh_xadm_cplh_cdts, xtlh_xadm_cpld_cdts}; If the number of advertised completion credits (both header and data) are infinite, then the default would be {12'b0, 8'hFF, 12'hFFF}.

Address: 1FF_C000h base + 738h offset = 1FF_C738h



PCIE_PL_TCFCSR field descriptions

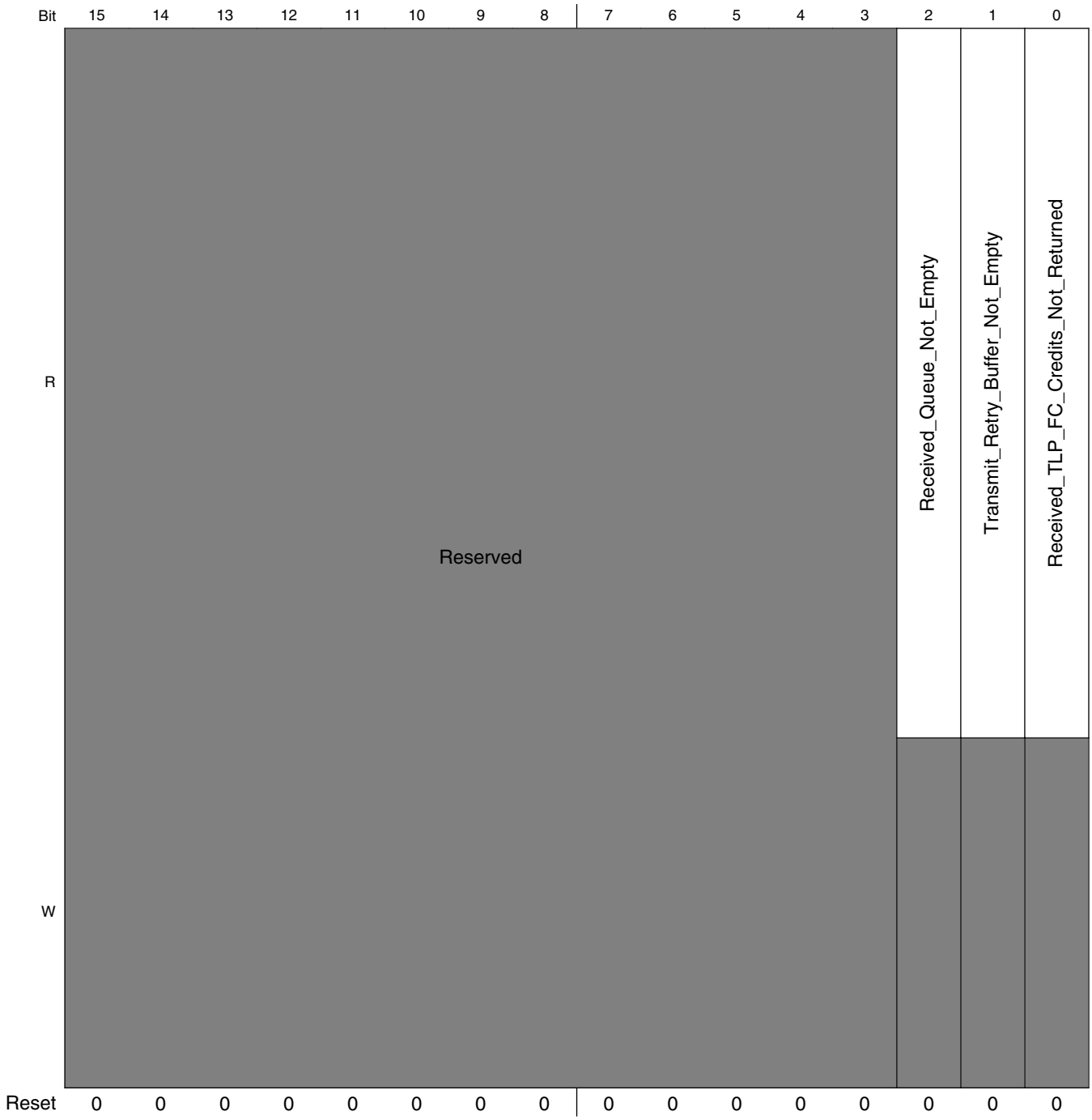
Field	Description
31–20 -	This field is reserved. Reserved
19–12 Transmit_Completion_Header_FC_Credits	Transmit Completion Header FC Credits The Completion Header credits advertised by the receiver at the other end of the Link, updated with each UpdateFC DLLP.
Transmit_Completion_Data_FC_Credits	Transmit Completion Data FC Credits The Completion Data credits advertised by the receiver at the other end of the Link, updated with each UpdateFC DLLP.

48.9.16 Queue Status Register (PCIE_PL_QSR)

Offset: 0x700 + 0x3C

Address: 1FF_C000h base + 73Ch offset = 1FF_C73Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



PCIE_PL_QSR field descriptions

Field	Description
31–3 -	This field is reserved. Reserved
2 Received_ Queue_Not_ Empty	Received Queue Not Empty Indicates there is data in one or more of the receive buffers.

Table continues on the next page...

PCIE_PL_QSR field descriptions (continued)

Field	Description
1 Transmit_Retry_Buffer_Not_Empty	Transmit Retry Buffer Not Empty Indicates that there is data in the transmit retry buffer.
0 Received_TLP_FC_Credits_Not_Returned	Received TLP FC Credits Not Returned Indicates that the core has sent a TLP but has not yet received an UpdateFC DLLP indicating that the credits for that TLP have been restored by the receiver at the other end of the Link. **Note**: This bit is for simulation only and will always be synthesized as 0.

48.9.17 VC Transmit Arbitration Register 1 (PCIE_PL_VCTAR1)

Offset: 0x700 + 0x40

VC Transmit Arbitration Registers 1 and 2 specify the weights assigned to VC0-VC7 to be used for WRR transmit arbitration for VCs in the LPVC group. The following rules and restrictions apply regarding the values programmed in VC Transmit Arbitration Registers 1 and 2:

- There are 8 bits allocated for each weight value.
- No weight value for a VC in the LPVC group can be less than 1.
- No weight value can be greater than the number of phases in the selected arbitration scheme.
- The sum of the weights assigned to all VCs in the LPVC group must equal the number of phases in the selected arbitration scheme. For example, if 64-phase WRR arbitration is selected, the total of all WRR Weight values for all VCs in the LPVC group must equal 64.

Each of the VC numbers listed in the bit field table is a VC ID, not the VC structure number. VC Transmit Arbitration Registers 1 and 2 are hardwired to the default values set by the configuration parameters listed in the Default columns of the bit field table.

Address: 1FF_C000h base + 740h offset = 1FF_C740h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WRR_Weight_for_VC3								WRR_Weight_for_VC2								WRR_Weight_for_VC1								WRR_Weight_for_VC0							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

PCIE_PL_VCTAR1 field descriptions

Field	Description
31–24 WRR_Weight_ for_VC3	WRR Weight for VC3
23–16 WRR_Weight_ for_VC2	WRR Weight for VC2
15–8 WRR_Weight_ for_VC1	WRR Weight for VC1
WRR_Weight_ for_VC0	WRR Weight for VC0

48.9.18 VC Transmit Arbitration Register 2 (PCIE_PL_VCTAR2)

Offset: 0x700 + 0x44

VC Transmit Arbitration Registers 1 and 2 specify the weights assigned to VC0-VC7 to be used for WRR transmit arbitration for VCs in the LPVC group. The following rules and restrictions apply regarding the values programmed in VC Transmit Arbitration Registers 1 and 2:

- There are 8 bits allocated for each weight value.
- No weight value for a VC in the LPVC group can be less than 1.
- No weight value can be greater than the number of phases in the selected arbitration scheme.
- The sum of the weights assigned to all VCs in the LPVC group must equal the number of phases in the selected arbitration scheme. For example, if 64-phase WRR arbitration is selected, the total of all WRR Weight values for all VCs in the LPVC group must equal 64.

Each of the VC numbers listed in the bit field table is a VC ID, not the VC structure number. VC Transmit Arbitration Registers 1 and 2 are hardwired to the default values set by the configuration parameters listed in the Default columns of the bit field table.

Address: 1FF_C000h base + 744h offset = 1FF_C744h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WRR_Weight_for_VC7								WRR_Weight_for_VC6								WRR_Weight_for_VC5								WRR_Weight_for_VC4							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_PL_VCTAR2 field descriptions

Field	Description
31–24 WRR_Weight_ for_VC7	WRR Weight for VC7
23–16 WRR_Weight_ for_VC6	WRR Weight for VC6
15–8 WRR_Weight_ for_VC5	WRR Weight for VC5
WRR_Weight_ for_VC4	WRR Weight for VC4

48.9.19 VC0 Posted Receive Queue Control (PCIE_PL_VC0PRQC)

NOTE

The data and header credits fields of the Receive Queue Control registers are used in all receive buffer configurations.

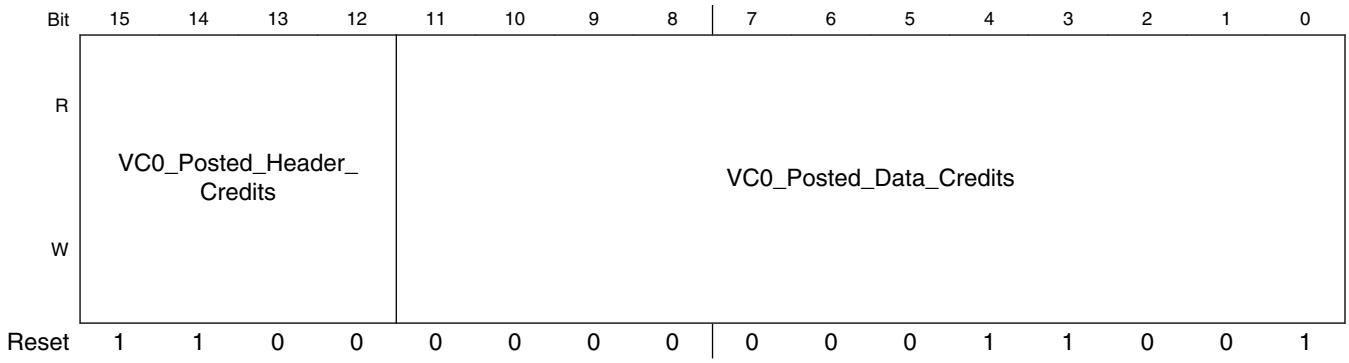
NOTE

All other fields of the Receive Queue Control and Depth registers are used only in the segmented- buffer configuration.

Offset: 0x700 + 0x48

Address: 1FF_C000h base + 748h offset = 1FF_C748h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	VC_Ordering_for_ Receive_Queues	TLP_Type_Ordering_for_ VC0	Reserved						VC0_Posted_TLP_ Queue_Mode			Reserved	VC0_Posted_Header_ Credits			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0



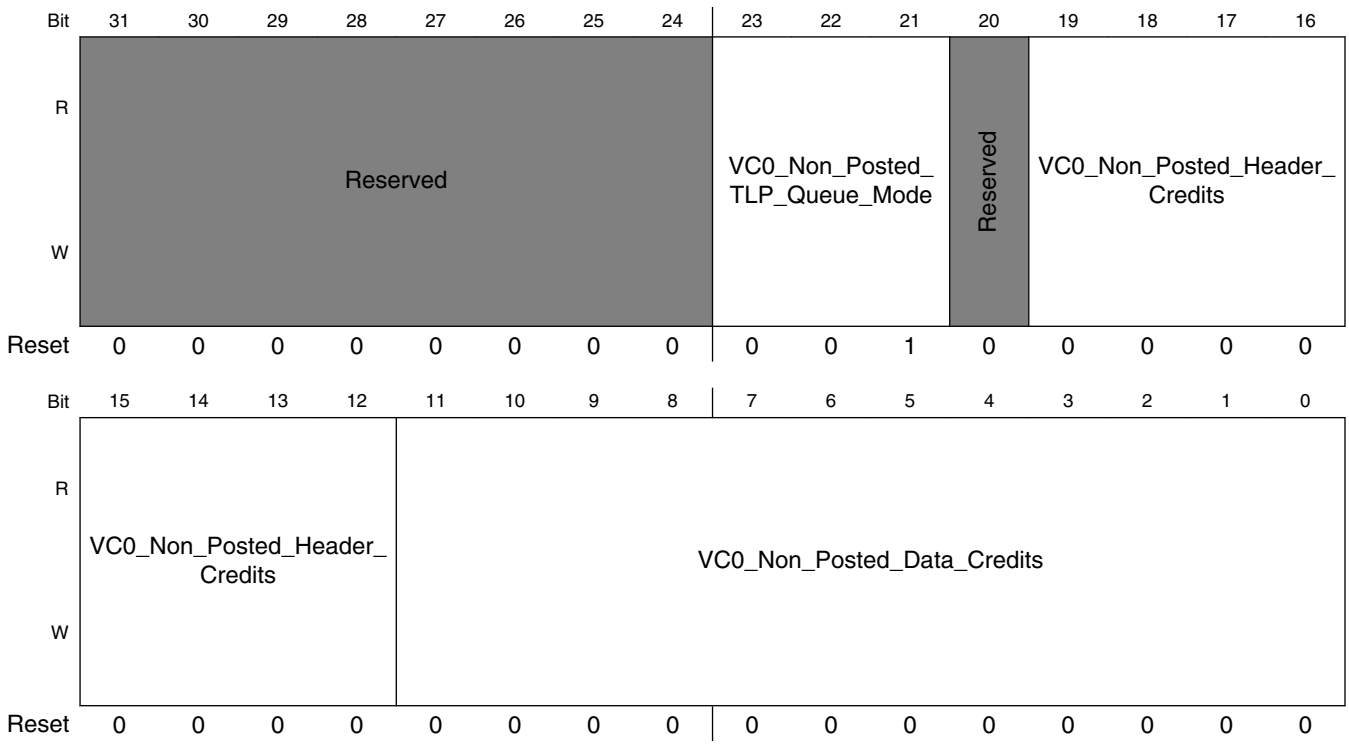
PCIE_PL_VC0PRQC field descriptions

Field	Description
31 VC_Ordering_for_Receive_Queues	VC Ordering for Receive Queues Determines the VC ordering rule for the receive queues, used only in the segmented-buffer configuration, writable through the DBI: 1 Strict ordering, higher numbered VCs have higher priority 0 Round robin
30 TLP_Type_Ordering_for_VC0	TLP Type Ordering for VC0 Determines the TLP type ordering rule for VC0 receive queues, used only in the segmented-buffer configuration, writable through the DBI: 1 Ordering of received TLPs follows the rules in PCI Express 3.0 Specification. 0 Strict ordering for received TLPs: Posted, then Completion, then Non-Posted
29–24 -	This field is reserved. Reserved
23–21 VC0_Posted_TLP_Queue_Mode	VC0 Posted TLP Queue Mode The operating mode of the Posted receive queue for VC0, used only in the segmented-buffer configuration, writable through the DBI. Only one bit can be set at a time: • Bit 23: Bypass • Bit 22: Cut-through • Bit 21: Store-and-forward
20 -	This field is reserved. Reserved
19–12 VC0_Posted_Header_Credits	VC0 Posted Header Credits The number of initial Posted header credits for VC0, used for all receive queue buffer configurations. This field is not writable through the DBI
VC0_Posted_Data_Credits	VC0 Posted Data Credits The number of initial Posted data credits for VC0, used for all receive queue buffer configurations. This field is not writable through the DBI

48.9.20 VC0 Non-Posted Receive Queue Control (PCIE_PL_VC0NRQC)

Offset: 0x700 + 0x4C

Address: 1FF_C000h base + 74Ch offset = 1FF_C74Ch



PCIE_PL_VC0NRQC field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–21 VC0_Non_Posted_TLP_Queue_Mode	VC0 Non-Posted TLP Queue Mode The operating mode of the Non-Posted receive queue for VC0, used only in the segmented-buffer configuration, writable through the DBI. Only one bit can be set at a time: • Bit 23: Bypass • Bit 22: Cut-through • Bit 21: Store-and-forward
20 -	This field is reserved. Reserved
19–12 VC0_Non_Posted_Header_Credits	VC0 Non-Posted Header Credits The number of initial Non-Posted header credits for VC0, used for all receive queue buffer configurations. This field is not writable through the DBI

Table continues on the next page...

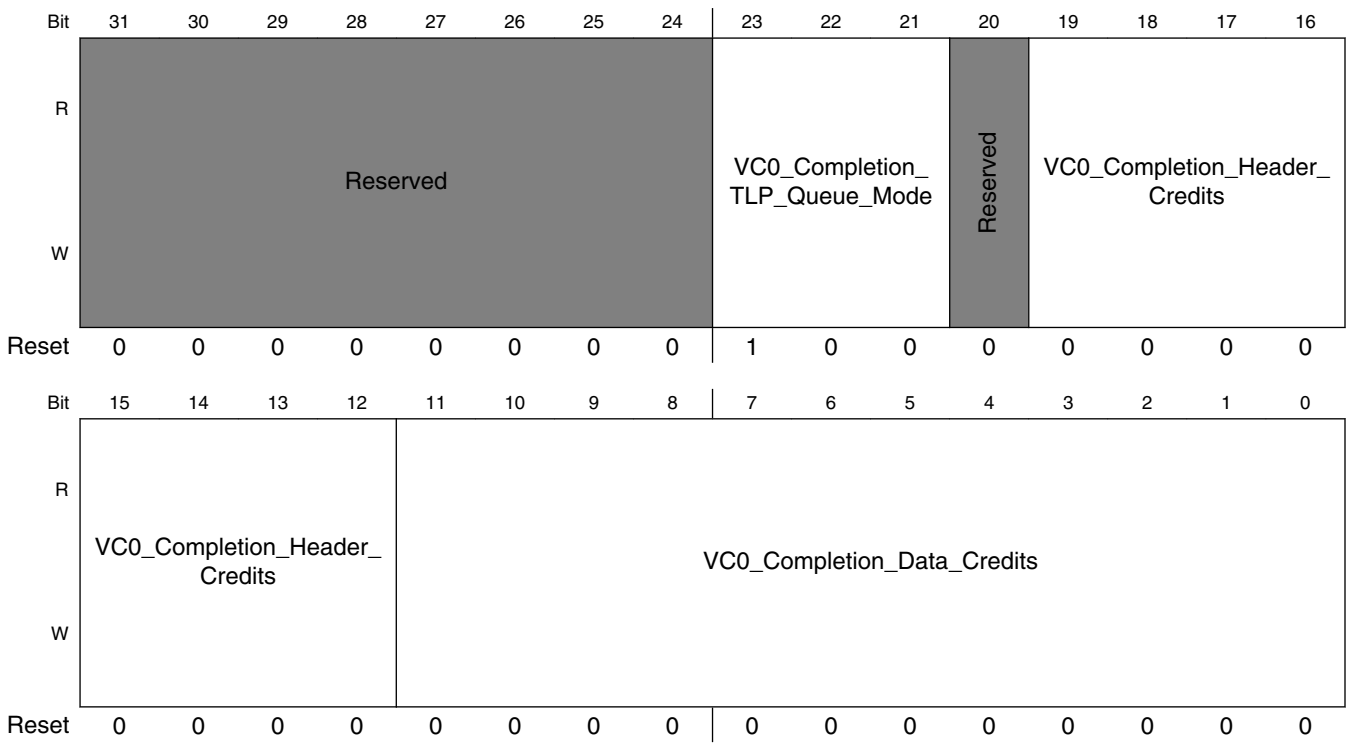
PCIE_PL_VC0NRQC field descriptions (continued)

Field	Description
VC0_Non_ Posted_Data_ Credits	VC0 Non-Posted Data Credits The number of initial Non-Posted data credits for VC0, used for all receive queue buffer configurations. This field is not writable through the DBI

48.9.21 VC0 Completion Receive Queue Control (PCIE_PL_VC0CRQC)

Offset: 0x700 + 0x50

Address: 1FF_C000h base + 750h offset = 1FF_C750h



PCIE_PL_VC0CRQC field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–21 VC0_ Completion_ TLP_Queue_ Mode	VC0 Completion TLP Queue Mode The operating mode of the Completion receive queue for VC0, used only in the segmented-buffer configuration, writable through the DBI. Only one bit can be set at a time: • Bit 23: Bypass

Table continues on the next page...

PCIE_PL_VC0CRQC field descriptions (continued)

Field	Description
	• Bit 22: Cut-through • Bit 21: Store-and-forward
20 -	This field is reserved. Reserved
19–12 VC0_ Completion_ Header_Credits	VC0 Completion Header Credits The number of initial Completion header credits for VC0, used for all receive queue buffer configurations. This field is not writable through the DBI
VC0_ Completion_ Data_Credits	VC0 Completion Data Credits The number of initial Completion data credits for VC0, used for all receive queue buffer configurations. This field is not writable through the DBI.

48.9.22 VCn Posted Receive Queue Control (PCIE_PL_VCnPRQC)

Offset: 0x700 + 0x48 + C*n (n=[1:7])

Address: 1FF_C000h base + 754h offset = 1FF_C754h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved	TLP_Type_Ordering_for_VC1	Reserved						VC1_Posted_TLP_Queue_Mode			Reserved	VC1_Posted_Header_Credits			
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VC1_Posted_Header_Credits				VC1_Posted_Data_Credits											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

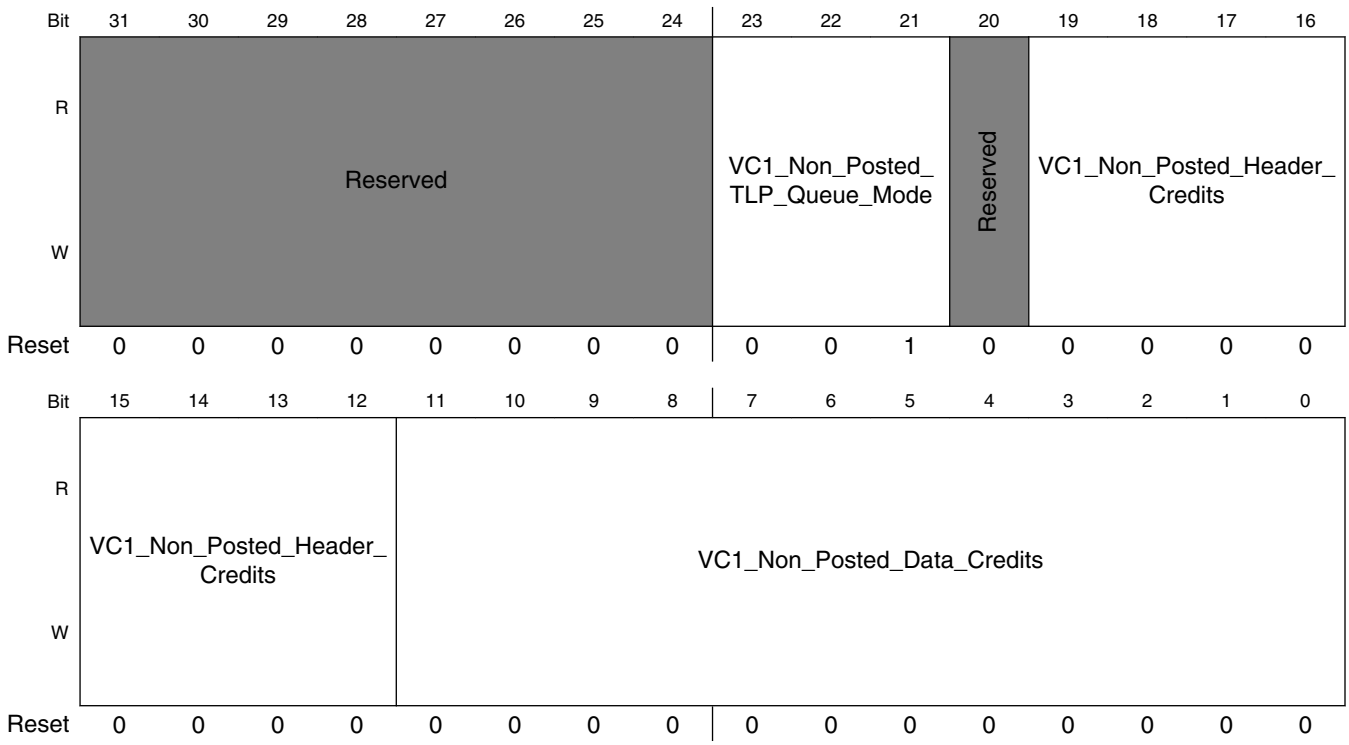
PCIE_PL_VCnPRQC field descriptions

Field	Description
31 -	This field is reserved. Reserved
30 TLP_Type_Ordering_for_VC1	TLP Type Ordering for VC1 Determines the TLP type ordering rule for VC1 receive queues, used only in the segmented-buffer configuration, writable through the DBI: • 1: Ordering of received TLPs follows the rules in *PCI Express Base 3.0 Specification* • 0: Strict ordering for received TLPs: Posted, then Completion, then Non-Posted
29–24 -	This field is reserved. Reserved
23–21 VC1_Posted_TLP_Queue_Mode	VC1 Posted TLP Queue Mode The operating mode of the Posted receive queue for VC1, used only in the segmented-buffer configuration, writable through the DBI. Only one bit can be set at a time: • Bit 23: Bypass • Bit 22: Cut-through • Bit 21: Store-and-forward
20 -	This field is reserved. Reserved
19–12 VC1_Posted_Header_Credits	VC1 Posted Header Credits The number of initial Posted header credits for VC1, used for all receive queue buffer configurations. This field is not writable through the DBI
VC1_Posted_Data_Credits	VC1 Posted Data Credits The number of initial Posted data credits for VC1, used for all receive queue buffer configurations. This field is not writable through the DBI

48.9.23 VCn Non-Posted Receive Queue Control (PCIE_PL_VCnNRQC)

Offset: 0x700 + 0x4C + C*n (n=[1:7])

Address: 1FF_C000h base + 758h offset = 1FF_C758h



PCIE_PL_VCnNRQC field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–21 VC1_Non_ Posted_TLP_ Queue_Mode	VC1 Non-Posted TLP Queue Mode The operating mode of the Non-Posted receive queue for VC1, used only in the segmented-buffer configuration, writable through the DBI. Only one bit can be set at a time: • Bit 23: Bypass • Bit 22: Cut-through • Bit 21: Store-and-forward
20 -	This field is reserved. Reserved
19–12 VC1_Non_ Posted_Header_ Credits	VC1 Non-Posted Header Credits The number of initial Non-Posted header credits for VC1, used for all receive queue buffer configurations. This field is not writable through the DBId.

Table continues on the next page...

PCIE_PL_VCnNRQC field descriptions (continued)

Field	Description
VC1_Non_ Posted_Data_ Credits	VC1 Non-Posted Data Credits The number of initial Non-Posted data credits for VC1, used for all receive queue buffer configurations. This field is not writable through the DBI

48.9.24 VCn Completion Receive Queue Control (PCIE_PL_VCnCRQC)

Offset: 0x700 + 0x50 + C*n (n=[1:7])

Address: 1FF_C000h base + 75Ch offset = 1FF_C75Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved								VC1_Completion_ TLP_Queue_Mode			Reserved	VC1_Completion_Header_ Credits			
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VC1_Completion_Header_ Credits				VC1_Completion_Data_Credits											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_PL_VCnCRQC field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–21 VC1_ Completion_ TLP_Queue_ Mode	VC1 Completion TLP Queue Mode The operating mode of the Completion receive queue for VC1, used only in the segmented-buffer configuration, writable through the DBI. Only one bit can be set at a time: • Bit 23: Bypass

Table continues on the next page...

PCIE_PL_VCnCRQC field descriptions (continued)

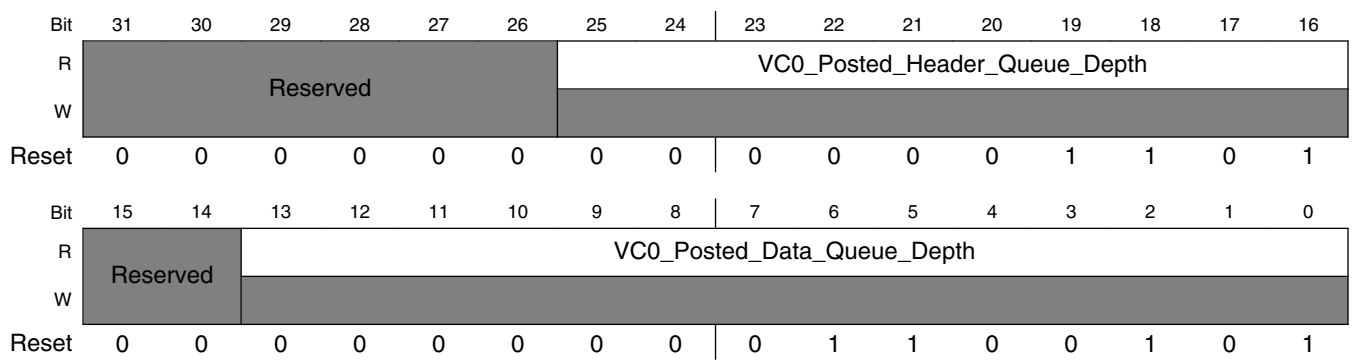
Field	Description
	• Bit 22: Cut-through • Bit 21: Store-and-forward
20 -	This field is reserved. Reserved
19–12 VC1_ Completion_ Header_Credits	VC1 Completion Header Credits The number of initial Completion header credits for VC1, used for all receive queue buffer configurations. This field is not writable through the DBI
VC1_ Completion_ Data_Credits	VC1 Completion Data Credits The number of initial Completion data credits for VC1, used for all receive queue buffer configurations. This field is not writable through the DBI

48.9.25 VC0 Posted Buffer Depth (PCIE_PL_VC0PBD)

- The Buffer Depth registers are used only in the segmented-buffer configuration.
- Writing to these registers is not possible (through the DBI)

Offset: 0x700 + 0xA8

Address: 1FF_C000h base + 7A8h offset = 1FF_C7A8h



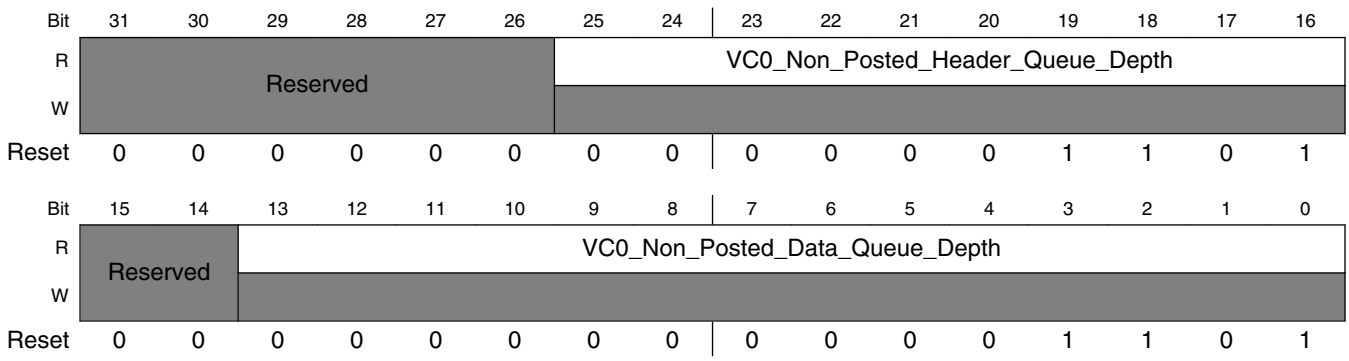
PCIE_PL_VC0PBD field descriptions

Field	Description
31–26 -	This field is reserved. Reserved
25–16 VC0_Posted_ Header_Queue_ Depth	VC0 Posted Header Queue Depth Sets the number of entries in the Posted header queue for VC0 when using the segmented-buffer configuration.Not writable through the DBI
15–14 -	This field is reserved. Reserved
VC0_Posted_ Data_Queue_ Depth	VC0 Posted Data Queue Depth Sets the number of entries in the Posted data queue for VC0 when using the segmented-buffer configuration.Not writable through the DBI

48.9.26 VC0 Non-Posted Buffer Depth (PCIE_PL_VC0NPBD)

Offset: 0x700 + 0xAC

Address: 1FF_C000h base + 7ACh offset = 1FF_C7ACh



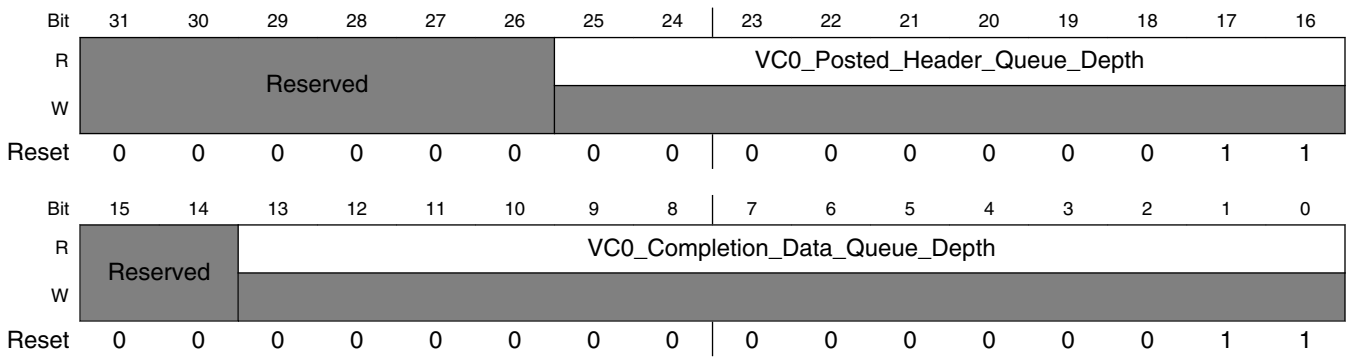
PCIE_PL_VC0NPBD field descriptions

Field	Description
31–26 -	This field is reserved. Reserved
25–16 VC0_Non_Posted_Header_Queue_Depth	VC0 Non-Posted Header Queue Depth Sets the number of entries in the Non-Posted header queue for VC0 when using the segmented-buffer configuration.Not writable through the DBI
15–14 -	This field is reserved. Reserved
VC0_Non_Posted_Data_Queue_Depth	VC0 Non-Posted Data Queue Depth Sets the number of entries in the Non-Posted data queue for VC0 when using the segmented-buffer configuration. Not writable through the DBI

48.9.27 VC0 Completion Buffer Depth (PCIE_PL_VC0CBD)

Offset: 0x700 + 0xB0

Address: 1FF_C000h base + 7B0h offset = 1FF_C7B0h



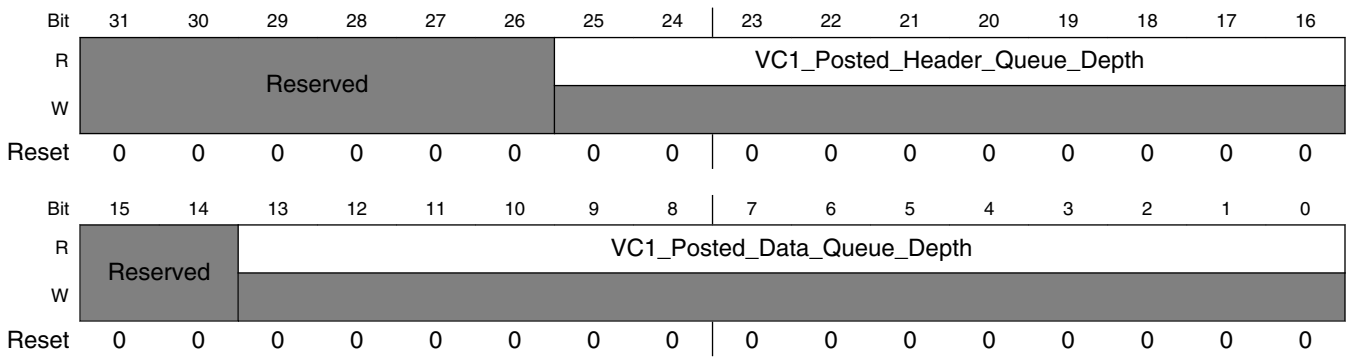
PCIE_PL_VC0CBD field descriptions

Field	Description
31–26 -	This field is reserved. Reserved
25–16 VC0_Posted_Header_Queue_Depth	VC0 Posted Header Queue Depth Sets the number of entries in the Completion header queue for VC0 when using the segmented-buffer configuration.Not writable through the DBI
15–14 -	This field is reserved. Reserved
VC0_Completion_Data_Queue_Depth	VC0 Completion Data Queue Depth Sets the number of entries in the Completion data queue for VC0 when using the segmented-buffer configuration. Not writable through the DBI

48.9.28 VCn Posted Buffer Depth (PCIE_PL_VC1PBD)

Offset: 0x700 + 0xA8 + C*n (n=[1:7])

Address: 1FF_C000h base + 7B4h offset = 1FF_C7B4h



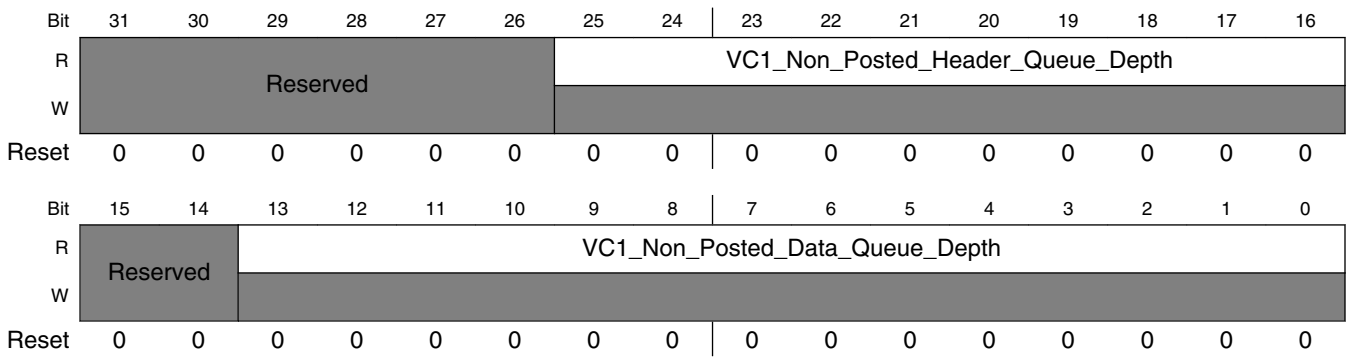
PCIE_PL_VC1PBD field descriptions

Field	Description
31–26 -	This field is reserved. Reserved
25–16 VC1_Posted_Header_Queue_Depth	VC1 Posted Header Queue Depth Sets the number of entries in the Posted header queue for VC1 when using the segmented-buffer configuration.Not writable through the DBI
15–14 -	This field is reserved. Reserved
VC1_Posted_Data_Queue_Depth	VC1 Posted Data Queue Depth Sets the number of entries in the Posted data queue for VC1 when using the segmented-buffer configuration.Not writable through the DBI

48.9.29 VCn Non-Posted Buffer Depth (PCIE_PL_VC1NPBD)

Offset: 0x700 + 0xAC + C*n (n=[1:7])

Address: 1FF_C000h base + 7B8h offset = 1FF_C7B8h



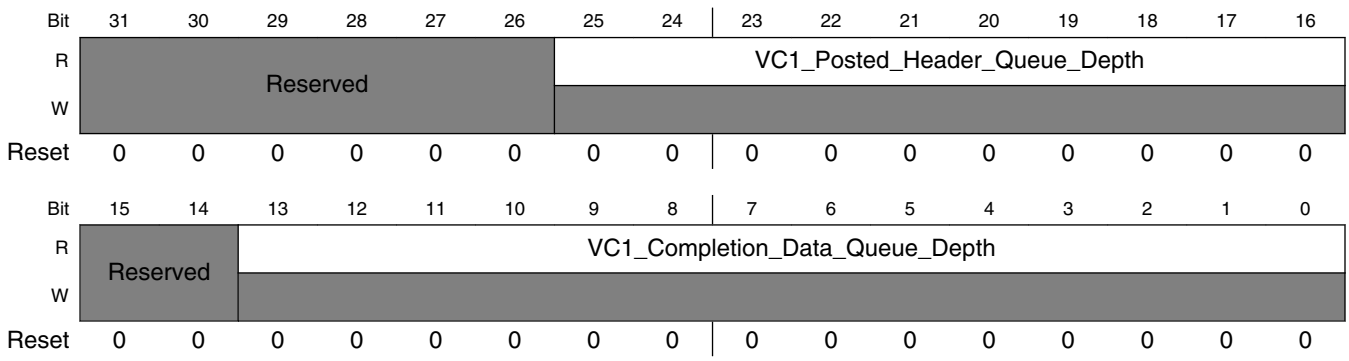
PCIE_PL_VC1NPBD field descriptions

Field	Description
31–26 -	This field is reserved. Reserved
25–16 VC1_Non_Posted_Header_Queue_Depth	VC1 Non-Posted Header Queue Depth Sets the number of entries in the Non-Posted header queue for VC1 when using the segmented-buffer configuration. Not writable through the DBI
15–14 -	This field is reserved. Reserved
VC1_Non_Posted_Data_Queue_Depth	VC1 Non-Posted Data Queue Depth Sets the number of entries in the Non-Posted data queue for VC1 when using the segmented-buffer configuration. Not writable through the DBI

48.9.30 VCnCompletion Buffer Depth (PCIE_PL_VC1CBD)

Offset: 0x700 + 0xB0 + C*n (n=[1:7])

Address: 1FF_C000h base + 7BCh offset = 1FF_C7BCh



PCIE_PL_VC1CBD field descriptions

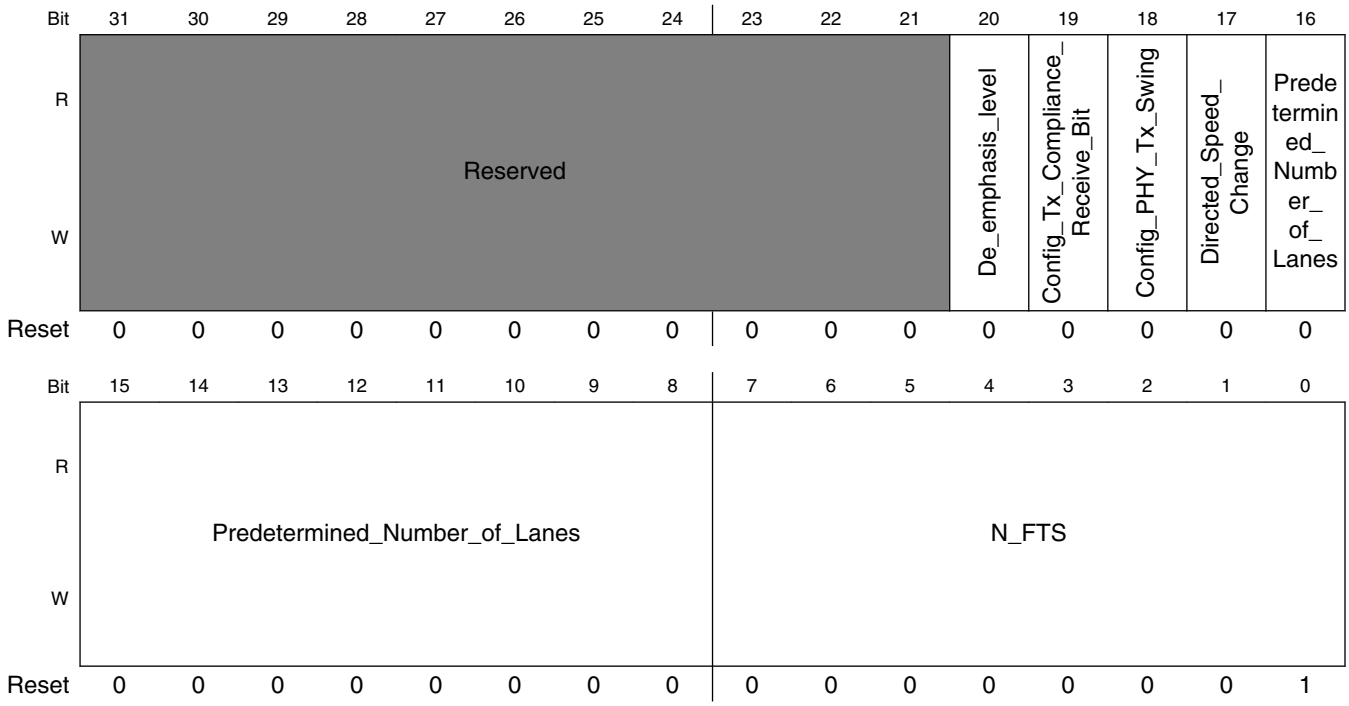
Field	Description
31–26 -	This field is reserved. Reserved
25–16 VC1_Posted_Header_Queue_Depth	VC1 Posted Header Queue Depth Sets the number of entries in the Completion header queue for VC1 when using the segmented-buffer configuration. Not writable through the DBI
15–14 -	This field is reserved. Reserved
VC1_Completion_Data_Queue_Depth	VC1 Completion Data Queue Depth Sets the number of entries in the Completion data queue for VC1 when using the segmented-buffer configuration. Not writable through the DBI

48.9.31 Gen2 Control Register (PCIE_PL_G2CR)

The Port Logic Gen2 Control Register controls features specific to data rates greater than 2.5 GT/s. The "Lane Enable" field is an exception in that it applies regardless of the data rate.

Offset: 0x700 + 0x10C

Address: 1FF_C000h base + 80Ch offset = 1FF_C80Ch



PCIE_PL_G2CR field descriptions

Field	Description
31–21 -	This field is reserved. Reserved
20 De_emphasis_level	Used to set the de-emphasis level for upstream ports.
19 Config_Tx_Compliance_Receive_Bit	Config Tx Compliance Receive Bit When set to 1, signals LTSSM to transmit TS ordered sets with the compliance receive bit assert (equal to 1).
18 Config_PHY_Tx_Swing	Config PHY Tx Swing Indicates the voltage level the PHY should drive. When set to 1, indicates Full Swing. When set to 0, indicates Low Swing
17 Directed_Speed_Change	Directed Speed Change Indicates to the LTSSM whether or not to initiate a speed change to Gen2
16–8 Predetermined_Number_of_Lanes	Predetermined Number of Lanes Used to limit the effective link width to ignore "broken" lanes that detect a receiver. Indicates the number of lanes to check for exit from Electrical Idle in POLLING.ACTIVE and L2.IDLE. It is possible that the LTSSM may detect a Receiver on a 'bad' or 'broken' lane during the Detect Sub-state. However, it is also possible that such a lane may also fail to exit Electrical Idle and therefore prevent a valid link from being configured. Encoding is as follows: 0x01 = 1 lane

Table continues on the next page...

PCIE_PL_G2CR field descriptions (continued)

Field	Description
N_FTS	Sets the Number of Fast Training Sequences (N_FTS) that the core advertises as its N_FTS during Gen2 Link training. This value is used to inform the Link partner about the PHY's ability to recover synchronization after a low power state. The number should be provided by the PHY vendor. **NOTE: Note**: Do not set N_FTS to zero; doing so can cause the LTSSM to go into the recovery state when exiting from L0s.

48.9.32 PHY Status (PCIE_PL_PHY_STATUS)

Offset: 0x700 + 0x110

Address: 1FF_C000h base + 810h offset = 1FF_C810h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PHY_Status																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_PL_PHY_STATUS field descriptions

Field	Description
PHY_Status	PHY Status Data received directly from the phy_cfg_status bus.

48.9.33 PHY Control (PCIE_PL_PHY_CTRL)

Offset: 0x700 + 0x114

Address: 1FF_C000h base + 814h offset = 1FF_C814h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PHY_Control																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_PL_PHY_CTRL field descriptions

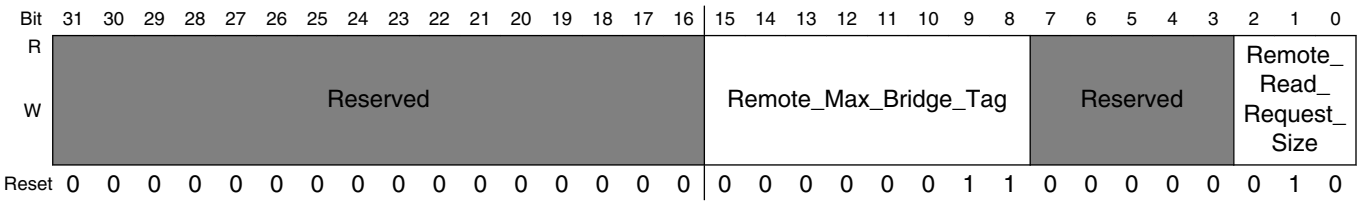
Field	Description
PHY_Control	PHY Control Data sent directly to the cfg_phy_control bus.

48.9.34 Master Response Composer Control Register 0 (PCIE_PL_MRCCR0)

You must not modify these registers for AHB configurations, as this feature is only supported for AXI.

Offset: 0x700 + 0x118

Address: 1FF_C000h base + 818h offset = 1FF_C818h



PCIE_PL_MRCCR0 field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
15–8 Remote_Max_ Bridge_Tag	Remote Max Bridge Tag Specifies the maximum number (-1) of Non-Posted AMBA requests outstanding at one time issued from the bridge master. Excludes any internally created TLP's as a result of decomposition. The core will automatically derive this when bits[2:0] (Remote Read Request Size) are written to. It is s saturated in core at CX_REMOTE_MAX_TAG since the TRGT_CPL_LUT has only this many entries. Therefore it is important that the core is initially sized (at configuration time pre-silicon) with the true maximum value of CX_REMOTE_MAX_TAG to take advantage of the ability to dynamically increase remote_max_bridge_tag from CX_REMOTE_MAX_BRIDGE_TAG to any new value up to a maximum of CX_REMOTE_MAX_TAG
7–3 -	This field is reserved. Reserved
Remote_Read_ Request_Size	Remote Read Request Size Specifies the largest amount of data (bytes) that will ever be requested (via an inbound MemRd TLP) by a remote device. Must never be programmed with a value that exceeds the value represented by the configuration parameter CX_REMOTE_RD_REQ_SIZE as the Master Response Composer RAM in the AXI bridge is sized using CX_REMOTE_RD_REQ_SIZE. Must only be programmed with the values 3'b000 to 3'b101. Any other value has the same effect as writing a value of 3'b000. Encoding is as follows: 000 128 001 256 010 512

Table continues on the next page...

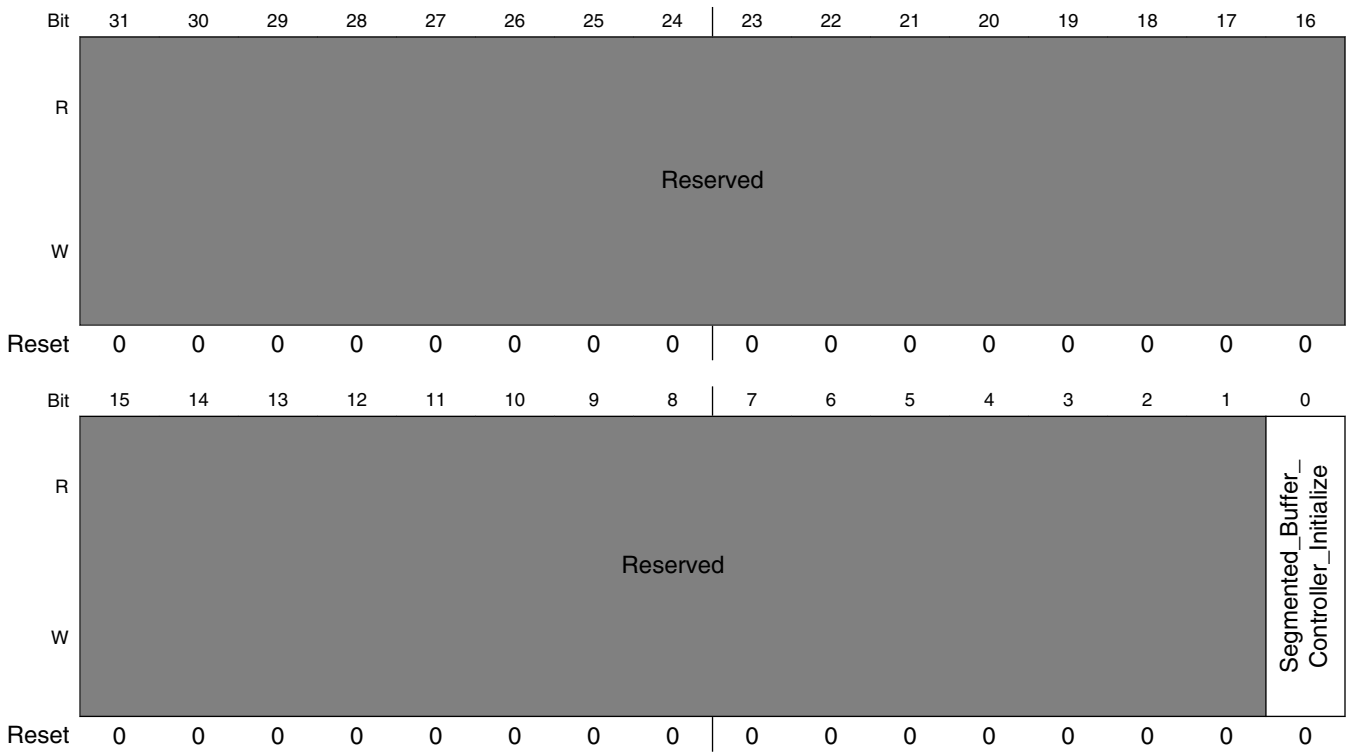
PCIE_PL_MRCCR0 field descriptions (continued)

Field	Description
	011 1024 100 2048 101 4096 default: 128

48.9.35 Master Response Composer Control Register 1 (PCIE_PL_MRCCR1)

Offset: 0x700 + 0x11C

Address: 1FF_C000h base + 81Ch offset = 1FF_C81Ch



PCIE_PL_MRCCR1 field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 Segmented_ Buffer_ Controller_ Initialize	Segmented Buffer Controller Initialize. Writing '1' to this (self-clearing register) causes any changes in the Master Response Composer Control Register 0 to take place in the bridge hardware. The sbc_init register triggers the initialization of the segmented buffer controller (DWC_sbc).

Table continues on the next page...

PCIE_PL_MRCCR1 field descriptions (continued)

Field	Description
	When sbc_init is written to, the segmented buffer controller (DWC_sbc) samples cfg_remote_max_bridge_tag and starts the internal finite state machine (FSM). * Reading from this self-clearing register field always returns a 0.

48.9.36 MSI Controller Address (PCIE_PL_MSICA)

See AHB/AXI MSI Controller (Optional in RC mode).

These registers are not part of the PCI Express MSI Capability Register structure which is detailed at MSI Capability Register Details.

Offset: 0x700 + 0x120

Address: 1FF_C000h base + 820h offset = 1FF_C820h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MSI_Controller_Address																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_PL_MSICA field descriptions

Field	Description
MSI_Controller_Address	MSI Controller Address System specified address for MSI memory write transaction termination. Within the AHB/AXI Bridge, every received Memory Write Request is examined to see if it targets the MSI Address that has been specified in the MSI Controller Address Register and also to see if it satisfies the definition of an MSI Interrupt Request. If these conditions are satisfied the Memory Write Request is marked as an MSI Request.

48.9.37 MSI Controller Upper Address (PCIE_PL_MSICUA)

Offset: 0x700 + 0x124

Address: 1FF_C000h base + 824h offset = 1FF_C824h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MSI_Controller_Upper_Address																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_PL_MSICUA field descriptions

Field	Description
MSI_Controller_ Upper_Address	MSI Controller Upper Address System specified upper address for MSI memory write transaction termination. Allows functions to support a 64- bit MSI address.

48.9.38 MSI Controller Interrupt n Enable (PCIE_PL_MSICIn_ENB)

Offset: 0x700 + 0x128 + C*n (n=[0:7])

Address: 1FF_C000h base + 828h offset = 1FF_C828h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MSI_Interrupt0_Enable																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_PL_MSICIn_ENB field descriptions

Field	Description
MSI_Interrupt0_ Enable	MSI Interrupt#0 Enable Specifies which interrupts are enabled. If an MSI is received from a disabled interrupt, no status bit gets set in MSI Controller Interrupt Status Register. Each bit corresponds to a single MSI Interrupt Vector.

48.9.39 MSI Controller Interrupt n Mask (PCIE_PL_MSICIn_MASK)

Offset: 0x700 + 0x12C + C*n (n=[0:7])

Address: 1FF_C000h base + 82Ch offset = 1FF_C82Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MSI_Interrupt0_Mask																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_PL_MSICIn_MASK field descriptions

Field	Description
MSI_Interrupt0_ Mask	MSI Interrupt#0 Mask

PCIE_PL_MSICIn_MASK field descriptions (continued)

Field	Description
	Allows enabled interrupts to be masked. If an MSI is received for a masked interrupt, the corresponding status bit gets set in the Interrupt Status Register but the msi_ctrl_int output is not set HIGH. Each bit corresponds to a single MSI Interrupt Vector.

48.9.40 MSI Controller Interrupt nStatus (PCIE_PL_MSICIn_STATUS)

Offset: 0x700 + 0x130 + C*n (n=[0:7])

Address: 1FF_C000h base + 830h offset = 1FF_C830h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MSI_Interrupt0_Status																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_PL_MSICIn_STATUS field descriptions

Field	Description
MSI_Interrupt0_Status	MSI Interrupt#0 Status If an MSI is detected for EP#0, one bit in this register is set. The decoding of the data payload of the MSI Memory Write Request determines which bit gets set. A status is bit is cleared by writing a 1 to the bit. Each bit corresponds to a single MSI Interrupt Vector.

48.9.41 MSI Controller General Purpose IO Register (PCIE_PL_MSICGPIO)

Offset: 0x700 + 0x188

Address: 1FF_C000h base + 888h offset = 1FF_C888h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MSI_GPIO_Register																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_PL_MSICGPIO field descriptions

Field	Description
MSI_GPIO_Register	MSI GPIO Register The contents of this register drives the top-level output msi_ctrl_io[31:0]

48.9.42 iATU Viewport Register (PCIE_PL_iATUVR)

See Internal Address Translation (iATU) for more information on iATU operation.

The iATU registers are programmed through an index (Viewport) register to reduce the footprint in the PCI Express Extended Configuration Space. The size of the required port logic space does not depend on the number of regions defined as the Viewport register is used to select which memory region is being accessed. There are 28 bytes of register space implemented *per address region* per direction. The number of address regions that are remapped by the iATU is 4 for inbound and 4 for outbound. However, only 32 bytes of the PCIe Extended Configuration Space Address Map is used.

Offset: 0x700 + 0x200

The viewport register has a "Region Direction" bit to determine whether an inbound or outbound region is being accessed and a "Region Index" field to determine which region to program/read when accessing the other address translation registers in the iATU Register Map below..

As an example, to access the Control, Base, Limit and Target registers for Outbound region number 4:

- Write 0x00000004 to Address {0x700 + 0x200} to index the Outbound Address Region #4
- Then proceed to write to any of the other registers in the iATU Register Map below.

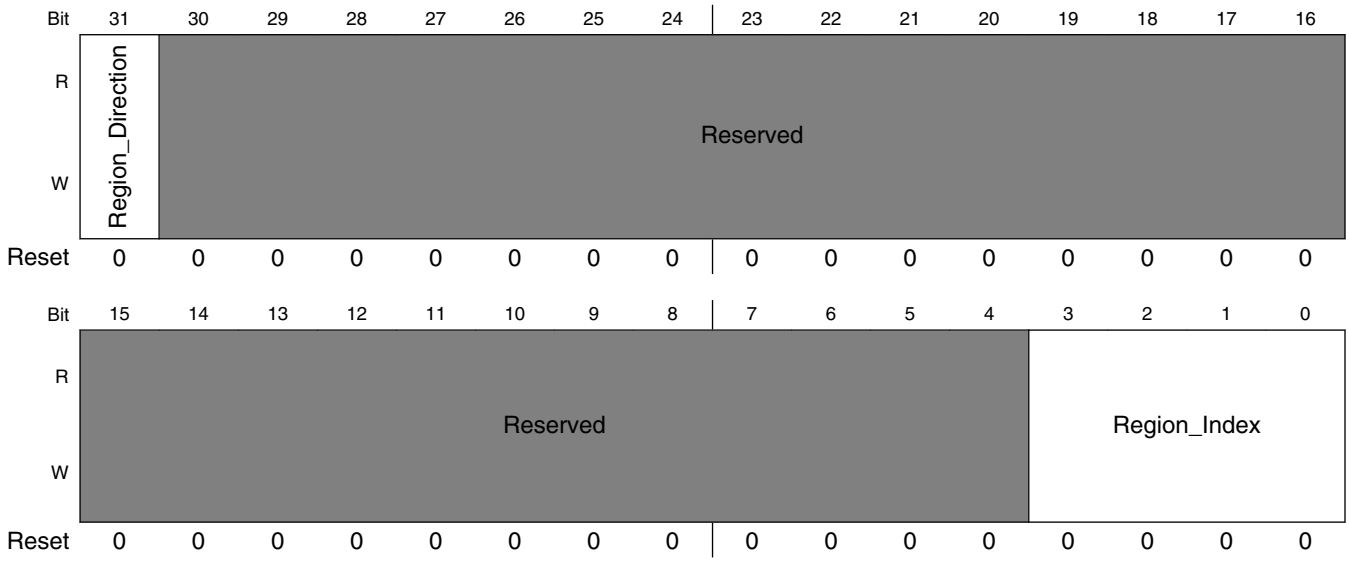
Table 48-162. iATU Register Map

Byte Offset	Description
+0x200	iATU Viewport Register
+0x204	iATU Region Control 1 Register
+0x208	iATU Region Control 2 Register
+0x20C	iATU Region Lower Base Address Register
+0x210	iATU Region Upper Base Address Register
+0x214	iATU Region Limit Address Register
+0x218	iATU Region Lower Target Address Register
+0x21C	iATU Region Upper Target Address Register

NOTE

Since AXI core is async to the core_clk, the iATU registers may not be updated while operations are in progress on the AXI Bridge Slave interface.

Address: 1FF_C000h base + 900h offset = 1FF_C900h



PCIE_PL_iATUVR field descriptions

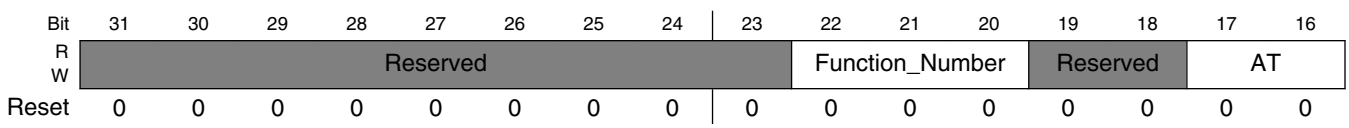
Field	Description
31 Region_Direction	Region Direction Defines the region being accessed as either 0 Outbound 1 Inbound
30–4 -	This field is reserved. Reserved
Region_Index	Region Index Defines which region is being accessed when writing to the control, base, limit and target registers. Must not be set to a number greater than CX_ATU_NUM_OUTBOUND_REGIONS - 1 when an outbound region is being accessed. Must not be set to a value greater than 3 since there are 4 regions for both inbound or outbound (4 each).

48.9.43 iATU Region Control 1 Register (PCIE_PL_iATURC1)

Offset: 0x700 + 0x204

footnote 1 - If all other enabled field-matches are successful

Address: 1FF_C000h base + 904h offset = 1FF_C904h



Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved					ATTR		TD	TC			TYPE				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_PL_iATURC1 field descriptions

Field	Description
31–23 -	This field is reserved. Reserved
22–20 Function_ Number	Function Number **Outbound**: When the address of an outbound TLP is matched to this region, then the function number used in generating the 'Function' part of the Requester ID (RID) field of the TLP is taken from this 3-bit register. The value in this register must be 0x0 **Inbound MEM/IO**: When the Address and BAR matching logic in the core indicate that a MEM/IO transaction matches a BAR in the function corresponding to this value, then address translation will proceed[1]. This check is only performed if the ìFunction Number Match Enable bit of the ìiATU Control 2 Register is set. **Inbound CFG0/CFG1**: When the destination function number as specified in the routing ID of the TLP header matches the function, then address translation will proceed1. This check is only performed if the ìFunction Number Match Enable bit of the ìiATU Control 2 Register is set.
19–18 -	This field is reserved. Reserved
17–16 AT	AT **NA**
15–11 -	This field is reserved. Reserved
10–9 ATTR	ATTR **Outbound**: When the address of an outbound TLP is matched to this region, then the ATTR field of the TLP is changed to the value in this register. **Inbound**: When the ATTR field of an inbound TLP is matched to this value, then address translation will proceed[1]. This check is only performed if the ìATTR Match Enable bit of the ìiATU Control 2 Register is set.
8 TD	TD **Outbound**: When the address of an outbound TLP is matched to this region, then the TD field of the TLP is changed to the value in this register. **Inbound**: When the TD field of an inbound TLP is matched to this value, then address translation will proceed1. This check is only performed if the ìTD Match Enable bit of the ìiATU Control 2 Register is set.
7–5 TC	TC **Outbound**: When the address of an outbound TLP is matched to this region, then the TC field of the TLP is changed to the value in this register. **Inbound**: When the TC field of an inbound TLP is matched to this value, then address translation will proceed[1]. This check is only performed if the ìTC Match Enable bit of the ìiATU Control 2 Register is set.
TYPE	TYPE **Outbound**: When the address of an outbound TLP is matched to this region, then the TYPE field of the TLP is changed to the value in this register.

Table continues on the next page...

PCIE_PL_iATURC1 field descriptions (continued)

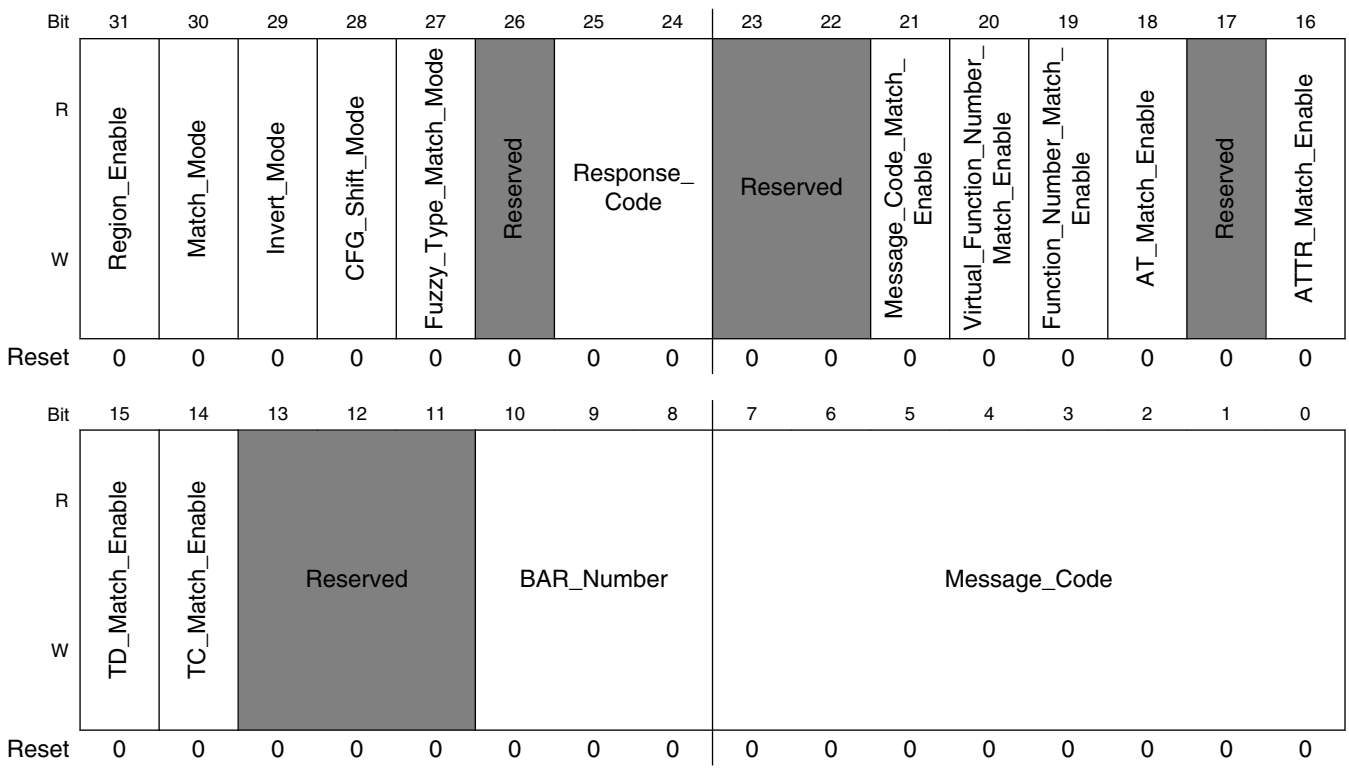
Field	Description
	Inbound: When the TYPE field of an inbound TLP is matched to this value, then address translation will proceed[2].

1. 1
2. 1

48.9.44 iATU Region Control 2 Register (PCIE_PL_iATURC2)

Offset: 0x700 + 0x208

Address: 1FF_C000h base + 908h offset = 1FF_C908h



PCIE_PL_iATURC2 field descriptions

Field	Description
31 Region_Enable	Region Enable **Outbound / Inbound**: This bit must be set to '1' for address translation to take place.
30 Match_Mode	Match Mode **Outbound**: Not used. **Inbound MEM/IO**: Determines Inbound matching mode for MEM/IO TLPs.

Table continues on the next page...

PCIE_PL_iATURC2 field descriptions (continued)

Field	Description
	• **0**: Address Mode. The iATU operates using addresses as in the Outbound direction. The Region Base and Limit Registers must be setup. • **1**: BAR Mode. BAR matching is used. The 'BAR Number' field is relevant. **Inbound CFG0**: Determines Inbound matching mode for CFG0 TLPs. • **0**: Routing ID match mode. The iATU interprets the Routing ID (Bytes 8 to 11 of TLP header) as an address. This corresponds to the upper 16 bits of the address in MEM/IO transactions. The Routing ID of the TLP must be within the base and limit of the iATU region for matching to proceed. • **1**: Accept Mode. The iATU accepts all CFG0 transactions as address matches. The routing ID in the CFG0 TLP is ignored. This is useful as all received CFG0 TLPs should be processed regardless of the Bus number. **Inbound MSG/MSGD**: Determines Inbound matching mode for MSG/MSGD TLPs. • **0**: Address Mode. The iATU treats the 3rd DWORD and 4th DWORD of the inbound MSG/MSGD TLP as an address and it is matched against the Region Base and Limit Registers. • **1**: Vendor ID match mode. This mode is relevant for ID-routed Vendor Defined Messages. The iATU ignores the Routing ID(Bus, Device, Function) in bits [31:16] of the 3rd DWORD of the TLP header, but matches against the Vendor ID in bits [15:0] of the 3rd DWORD of the TLP header. Bits [15:0] of the Region Upper Base register should be programmed with the required Vendor ID. The lower Base and Limit Register should be programmed to translate TLPs based on vendor specific information in the 4th DWORD of the TLP header.
29 Invert_Mode	Invert Mode **Outbound / Inbound**: When set the address matching region is inverted. Therefore, an address match occurs when the untranslated address is in the region outside the defined range (Base Address to Limit Address).
28 CFG_Shift_Mode	CFG Shift Mode This is useful for CFG transactions where the PCIe configuration mechanism maps bits [27:12] of the address to the bus/device and function number. This allows a CFG configuration space to be located in any 256MB window of the application memory space using a 28-bit effective address. **Outbound**: Shifts bits [27:12] of the untranslated address to form bits [31:16] of the translated address. **Inbound**: Shifts bits [31:16] of the untranslated address to form bits [27:12] of the translated address.
27 Fuzzy_Type_Match_Mode	Fuzzy Type Match Mode **Outbound**: Not used. **Inbound**: When enabled, the iATU relaxes the matching of the TLP TYPE field against the expected TYPE field so that • CfgRd0 and CfgRd1 TLPs are seen as identical. Similarly with CfgWr0 and CfgWr1. • MRd and MRdLk TLPs are seen as identical • The Routing field of Msg/MsgD TLPs is ignored For example, CFG0 in the TYPE field in the iATU Region Control 1 Register (PCIE_PL_iATURC1) will match against an inbound CfgRd0, CfgRd1, CfgWr0 or CfgWr1 TLP.
26 -	This field is reserved. Reserved
25–24 Response_Code	Response Code **Outbound**: Not used. **Inbound**: Defines the type of response to give for accesses matching this region. This overrides the normal RADM filter response.

Table continues on the next page...

PCIE_PL_iATURC2 field descriptions (continued)

Field	Description
	00 - Normal RADM filter response is used. 01 - Unsupported Request (UR) 10 - Completer Abort (CA) 11 - Not used / undefined / reserved.
23–22 -	This field is reserved. Reserved
21 Message_Code_Match_Enable	Message Code Match Enable **Outbound**: Not used. **Inbound**: Ensures that a successful Message Code TLP field comparison match occurs in MSG transactions (see Message Code field of the iATU Control 1 Register in PCIe CTRL Memory Map/Register Definition) for address translation to proceed.
20 Virtual_Function_Number_Match_Enable	Virtual Function Number Match Enable **Outbound**: Not used. **Inbound**: Ensures that a successful Virtual Function Number TLP field comparison match (see Virtual Function Number field of the iATU Control 1 Register in PCIe CTRL Memory Map/Register Definition occurs (in **MEM/IO** transactions) for address translation to proceed. **Note**: This bit must not be set at the same time as 'Function Number Match Enable'.
19 Function_Number_Match_Enable	Function Number Match Enable **Outbound**: Not used. **Inbound**: Ensures that a successful Function Number TLP field comparison match (see Function Number field of the iATU Control 1 Register in PCIe CTRL Memory Map/Register Definition) occurs (in **MEM/IO** and **CFG0/CFG1** transactions) for address translation to proceed. **Note**: This bit must not be set at the same time as 'Virtual Function Number Match Enable'.
18 AT_Match_Enable	AT Match Enable **Outbound**: Not used. **Inbound**: Ensures that a successful AT TLP field comparison match (see AT field of the iATU Control 1 Register in PCIe CTRL Memory Map/Register Definition) occurs for address translation to proceed. Only valid when the ATS_RX_ENABLE configuration parameter is 1.
17 -	This field is reserved. Reserved
16 ATTR_Match_Enable	ATTR Match Enable **Outbound**: Not used. **Inbound**: Ensures that a successful ATTR TLP field comparison match (see ATTR field of the PCIe CTRL Memory Map/Register Definition) occurs for address translation to proceed.
15 TD_Match_Enable	TD Match Enable **Outbound**: Not used. **Inbound**: Ensures that a successful TD TLP field comparison match (see TD field of the PCIe CTRL Memory Map/Register Definition) occurs for address translation to proceed.
14 TC_Match_Enable	TC Match Enable **Outbound**: Not used. **Inbound**: Ensures that a successful TC TLP field comparison match (see TC field of the PCIe CTRL Memory Map/Register Definition) occurs for address translation to proceed.

Table continues on the next page...

PCIE_PL_iATURC2 field descriptions (continued)

Field	Description
13–11 -	This field is reserved. Reserved
10–8 BAR_Number	BAR Number **Outbound**: Not used. **Inbound**: When the BAR number of an inbound **MEM** or **IO** TLP - that is matched by the normal internal BAR address matching mechanism - is the same as this field, address translation will proceed[1].This check is only performed if the ìMatch Mode bit of the iATU Region Control 2 Register (PCIE_PL_iATURC2) is set. IO translation would require either 00100b or 00101b in the inbound TLP TYPE; the BAR Number set in the range 000b - 101b and that BAR configured as an IO BAR. 000b - BAR#0 001b - BAR#1 010b - BAR#2 011b - BAR#3 100b - BAR#4 101b - BAR#5 110b - ROM 111b - reserved
Message_Code	Message Code **Outbound**: When the address of an outbound TLP is matched to this region, and the translated TLP TYPE field is Msg or MsgD; then the Message field of the TLP is changed to the value in this register. **Inbound**: When the TYPE field of an inbound **Msg/MsgD** TLP is matched to this value, then address translation will proceed[1]. This check is only performed if the ìFunction Message Code Match Enable bit of the iATU Region Control 2 Register (PCIE_PL_iATURC2) is set.

48.9.45 iATU Region Lower Base Address Register (PCIE_PL_iATURLBA)

The CX_ATU_MIN_REGION_SIZE configuration parameter (Value Range: 4 kB, 8 kB, 16 kB, 32 kB, 64 kB defaults to 64 kB) specifies the minimum size of an address translation region. For example, if set to 64 kB; the lower 16 bits of the Base, Limit and Target registers are zero and all address regions are aligned on 64 kB boundaries. More precisely, the lower log2(CX_ATU_MIN_REGION_SIZE) bits are zero.

Offset: 0x700 + 0x20C

Address: 1FF_C000h base + 90Ch offset = 1FF_C90Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Address_upper																Address_lower															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_PL_iATURLBA field descriptions

Field	Description
31–16 Address_upper	Forms bits [31:16] of the start address of the address region to be translated.
Address_lower	Forms bits [15:0] of the start address of the address region to be translated. The start address must be aligned to a CX_ATU_MIN_REGION_SIZE kB boundary, so these bits are always 0. A write to this location is ignored by the PCIe core.

48.9.46 iATU Region Upper Base Address Register (PCIE_PL_iATURUBA)

Offset: 0x700 + 0x210

Address: 1FF_C000h base + 910h offset = 1FF_C910h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Address																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_PL_iATURUBA field descriptions

Field	Description
Address	**Outbound / Inbound**: Forms bits [63:32] of the start (and end) address of the address region to be translated. **Outbound**: In systems with a 32-bit address space, this register is not used and therefore writing to this register has no effect.

48.9.47 iATU Region Limit Address Register (PCIE_PL_iATURLA)

Offset: 0x700 + 0x214

Address: 1FF_C000h base + 914h offset = 1FF_C914h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Address_upper																Address_lower															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

PCIE_PL_iATURLA field descriptions

Field	Description
31–16 Address_upper	Forms bits [31:16] of the end address of the address region to be translated.
Address_lower	Forms bits [15:0] of the end address of the address region to be translated. The end address must be aligned to a CX_ATU_MIN_REGION_SIZE kB boundary, so these bits are always 0. A write to this location is ignored by the PCIe core.

48.9.48 iATU Region Lower Target Address Register (PCIE_PL_iATURLTA)

Offset: 0x700 + 0x218

Address: 1FF_C000h base + 918h offset = 1FF_C918h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Address_upper																Address_lower															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_PL_iATURLTA field descriptions

Field	Description
31–16 Address_upper	Forms bits [31:16] of the of the new address of the translated region.
Address_lower	Forms bits [15:0] of the start address of the new address of the translated region. The start address must be aligned to a CX_ATU_MIN_REGION_SIZE kB boundary, so these bits are always 0. A write to this location is ignored by the PCIe core.

48.9.49 iATU Region Upper Target Address Register (PCIE_PL_iATURUTA)

Offset: 0x700 + 0x21C

Address: 1FF_C000h base + 91Ch offset = 1FF_C91Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Address																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_PL_iATURUTA field descriptions

Field	Description
Address	**Outbound / Inbound**: Forms bits [63:32] of the start address of the new address of the translated region. **Inbound**: In systems with a 32-bit address space, this register is not used and therefore writing to this register has no effect.

Chapter 49
PCI Express PHY (PCIe_PHY)

49.1 Overview

PCIe 2.0 PHY is a complete mixed-signal semiconductor intellectual property (IP) solution, designed for single-chip integration into computer applications.

The PCIe 2.0 PHY supports both the 5 Gbp/s data rate of the PCI Express Gen 2.0 specifications as well as being backwards compatible to the 2.5Gb/s Gen 1.1 specification.

This chapter provides an introduction to the PCIe 2.0 PHY and its features.

49.2 Applications

>

Designed for low power, the PCIe 2.0 PHY allows designers to introduce competitive products using the latest generation of the PCI Express standard.

49.3 PCIe2 PHY Features

49.3.1 Standards Compliance

The PCIe2 PHY is fully compliant with all of the required features of the following standards:

- PCI Express Base Specification, Revision 2.0 (including legacy 2.5-Gbps support
- 5.0 Gbps data rate

- PCI Express Base Specification, Revision 1.1
- 2.5Gbps data rate

49.3.2 PHY Features

- 5 Gbps data transmission rate
- Integrated PHY includes transmitter, receiver, PLL, digital core, and ESD.
- Programmable RX equalization
- Designed for excellent performance margin and receiver sensitivity
- Robust PHY architecture tolerates wide process, voltage and temperature variations
- Low-jitter PLL technology with excellent supply isolation
- IEEE 1149.6 (JTAG) boundary scan
- Built-in Self-Test (BIST) features for production, at-speed, testing on any digital tester
- 5Gb/s PCIe Gen 2 and 2.5Gb/s PCIe Gen 1.1 test modes supported
- Advanced built-in diagnostics including on-chip sampling scope for easy debug
- Visibility & controllability of hard macro functionality thru programmable registers in the design
- Over-rides on all ASIC side inputs for easy debug
- Access register space thru simple 16 bit parallel interface
- Access register space thru JTAG

49.4 Functional Description

This section provides a complete functional description of the block.

49.4.1 Clocks and Resets

49.4.1.1 Reference Clock Enables

To enable to lowest possible power state, there is an enable on the reference clock input buffer. When totally powered down and prior to removing the reference clock, can be de-asserted to completely shut down the PHY.

- To enable the reference clock buffer in the PCIe2 PHY, the signal phy_ref_ssp_en must be asserted after the reference clock is up and stable and prior to de-asserting phy_reset.

49.4.1.2 Reference Clock Frequency Selection

The MPLL in the PCIe2 PHY has the ability to multiply the reference clock by integer and non-integer values, allowing for a wide range input reference clock frequencies.

Based on the incoming reference clock frequency, the MPLL controls must be set as shown in the table below to get proper 5Gb/s operation. With these MPLL settings, the reference clock output phy_ref_output_clk will provide the frequencies shown in the right-most column the table below.

Table 49-1. Reference Clock Frequency Selection

Reference Clock (MHz)	Required Multiplication	ref_clkdiv2	mpll_multiplier[6:0]	phy_ref_output_clk
100	25	0	0011001	100 MHz
125	40	1	0101000	62.5 MHz
200	25	1	0011001	100 MHz

49.4.1.3 Spread Spectrum Clocking

The PCIe2 PHY uses the PCIe reference clock which may or may not be spread. Since the PCIe specification does not allow independently spread clocks, the ability of the PHY to add a spread to a fixed frequency reference clock must be disabled.

49.4.2 Termination Resistance Tuning

The PHY uses an external resistor to calibrate the termination impedances of the high speed inputs and outputs of the PHY.

49.4.3 Control Register Access

The CR port is a simple 16bit data/16bit address parallel port that is provided for on-chip access to the control registers inside the PCIe2 PHY.

While access to these registers is NOT required for normal operation of the PHY, this interface is included for users that want to access some of the diagnostic features of the PHY during normal operation or over-ride some of the basic PHY control signals.

This interface is completely asynchronous using a hand shake between phy_cr_cap_addr, phy_cr_cap_data, phy_cr_read, and phy_cr_write input commands with phy_cr_ack acknowledgements and phy_cr_data_out outputs from the PHY. The CR port access is broken down into Address, Read, and Write transactions.

The details of the Control Registers themselves is detailed Control Registers.

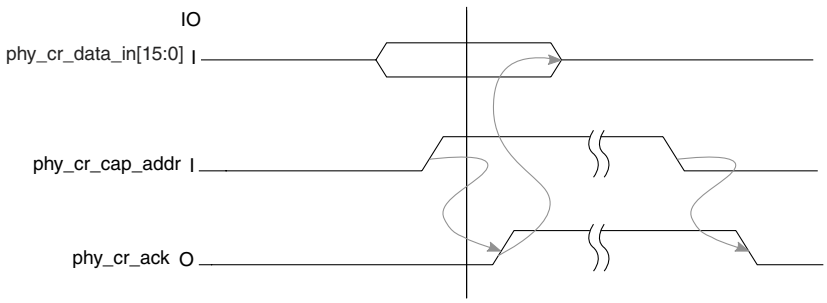


Figure 49-1. CR Bus Address Capture Transaction

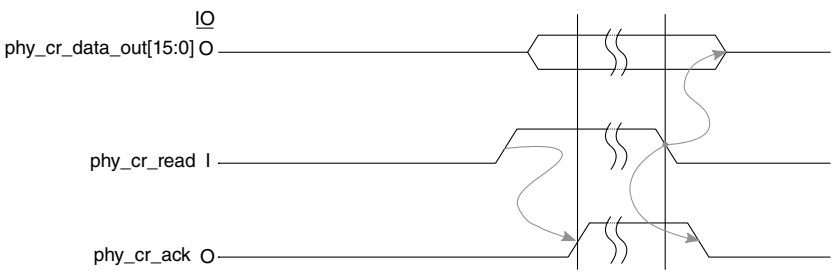


Figure 49-2. CR Bus Read Transaction

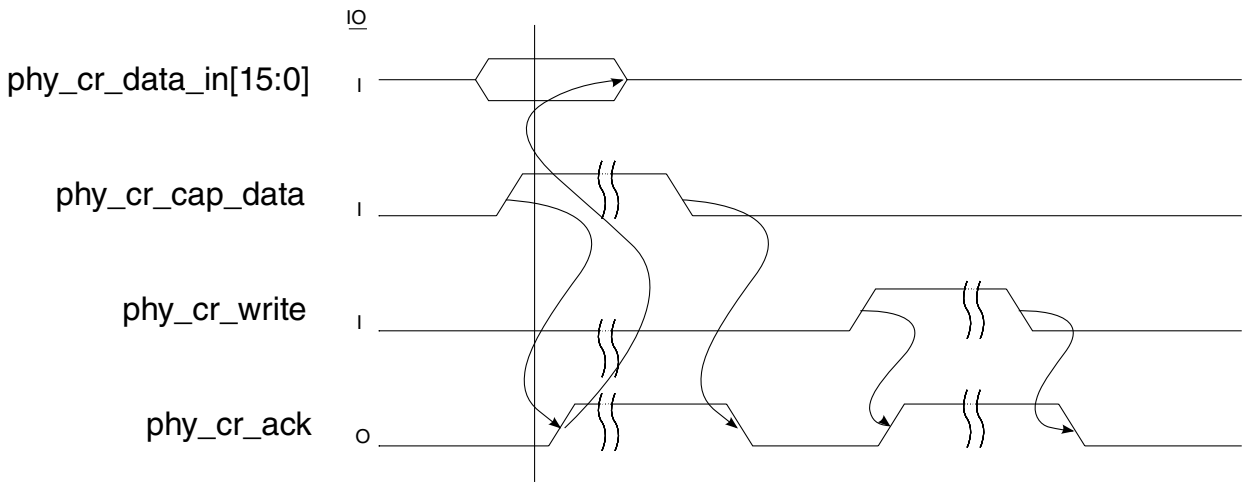


Figure 49-3. CR Bus Write Transaction

NOTE

In all cases, phy_cr_ack will de-assert when the command (phy_cr_cap_addr, phy_cr_cap_data, phy_cr_read, cr_write) is removed.

If the Control Register port is not going to be used, tie the input signals as follows,

- phy_cr_cap_addr =1'b0
- phy_cr_cap_data =1'b0
- phy_cr_data_in = 16'b0
- phy_cr_read = 1'b0
- phy_cr_write = 1'b0
- No connect (leave floating) the following:
- phy_cr_data_out[15:0]
- phy_cr_ack.

49.5 Control Memory Map/Register Definition

NOTE

PCIe PHY registers are only accessible by the corresponding controller (PCIE_PHY_CTRL_R and PCIE_PHY_STS_R) or in debug through the JTAG port. PCIe PHY is not memory mapped to processor address space, so the absolute addresses shown is the relative address and is not valid.

PCIE_PHY memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Register ID Low 16 bits (PCIE_PHY_IDCODE_LO)	16	R	0000h	49.5.1/4377
1	Register ID High 16 bits (PCIE_PHY_IDCODE_HI)	16	R	0000h	49.5.2/4377
2	Debug Register (PCIE_PHY_DEBUG)	16	R/W	000Ah	49.5.3/4378
3	Debug Register (PCIE_PHY_RTUNE_DEBUG)	16	R/W	0000h	49.5.4/4378
4	PCIE_PHY_RTUNE_STAT	16	R/W	0000h	49.5.5/4379
5	PCIE_PHY_SS_PHASE	16	R/W	0000h	49.5.6/4379
6	PCIE_PHY_SS_FREQ	16	R/W	3327h	49.5.7/4380
10	PCIE_PHY_ATEOVRD	16	R/W	0000h	49.5.8/4380
11	PCIE_PHY_MPLL_OVRD_IN_LO	16	R/W	004Ch	49.5.9/4381
11	PCIE_PHY_MPLL_OVRD_IN_HI	16	R/W	004Ch	49.5.10/ 4382
13	PCIE_PHY_SSC_OVRD_IN	16	R/W	0000h	49.5.11/ 4383
14	PCIE_PHY_BS_OVRD_IN	16	R/W	0000h	49.5.12/ 4384
15	PCIE_PHY_LEVEL_OVRD_IN	16	R/W	0000h	49.5.13/ 4385
16	PCIE_PHY_SUP_OVRD_OUT	16	R/W	0101h	49.5.14/ 4385
17	PCIE_PHY_MPLL_ASIC_IN	16	R	0000h	49.5.15/ 4386
18	PCIE_PHY_BS_ASIC_IN	16	R	0000h	49.5.16/ 4387
19	PCIE_PHY_LEVEL_ASIC_IN	16	R	0000h	49.5.17/ 4388
1A	PCIE_PHY_SSC_ASIC_IN	16	R	0000h	49.5.18/ 4389
1B	PCIE_PHY_SUP_ASIC_OUT	16	R	0000h	49.5.19/ 4389
1C	PCIE_PHY_ATEOVRD_STATUS	16	R	0000h	49.5.20/ 4390
20	PCIE_PHY_SCOPE_ENABLES	16	R/W	0000h	49.5.21/ 4391
21	PCIE_PHY_SCOPE_SAMPLES	16	R/W	0100h	49.5.22/ 4392
22	PCIE_PHY_SCOPE_COUNT	16	R/W	FFFFh	49.5.23/ 4392
23	PCIE_PHY_SCOPE_CTL	16	R/W	0000h	49.5.24/ 4393
24	PCIE_PHY_SCOPE_MASK_000	16	R/W	0000h	49.5.25/ 4393
25	PCIE_PHY_SCOPE_MASK_001	16	R/W	0000h	49.5.25/ 4393

Table continues on the next page...

PCIE_PHY memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
26	PCIE_PHY_SCOPE_MASK_010	16	R/W	0000h	49.5.25/ 4393
27	PCIE_PHY_SCOPE_MASK_011	16	R/W	0000h	49.5.25/ 4393
28	PCIE_PHY_SCOPE_MASK_100	16	R/W	0000h	49.5.25/ 4393
29	PCIE_PHY_SCOPE_MASK_101	16	R/W	0000h	49.5.25/ 4393
2A	PCIE_PHY_SCOPE_MASK_110	16	R/W	0000h	49.5.25/ 4393
2B	PCIE_PHY_SCOPE_MASK_111	16	R/W	0000h	49.5.25/ 4393
30	PCIE_PHY_MPLL_LOOP_CTL	16	R/W	00C0h	49.5.26/ 4394
32	PCIE_PHY_MPLL_ATB_MEAS2	16	R/W	0000h	49.5.27/ 4394
33	PCIE_PHY_MPLL_OVR	16	R/W	0000h	49.5.28/ 4395
34	PCIE_PHY_RTUNE_RTUNE_CTRL	16	R/W	0000h	49.5.29/ 4396
1000	PCIE_PHY_TX_OVRD_IN_LO	16	R/W	0000h	49.5.30/ 4397
1001	PCIE_PHY_TX_OVRD_IN_HI	16	R/W	0000h	49.5.31/ 4399
1003	PCIE_PHY_TX_OVRD_DRV_LO	16	R/W	0000h	49.5.32/ 4400
1004	PCIE_PHY_TX_OVRD_OUT	16	R/W	0000h	49.5.33/ 4400
1005	PCIE_PHY_RX_OVRD_IN_LO	16	R/W	0000h	49.5.34/ 4401
1006	PCIE_PHY_RX_OVRD_IN_HI	16	R/W	0000h	49.5.35/ 4402
1007	PCIE_PHY_RX_OVRD_OUT	16	R/W	0000h	49.5.36/ 4403
1008	PCIE_PHY_TX_ASIC_IN	16	R	0000h	49.5.37/ 4404
1009	PCIE_PHY_TX_ASIC_DRV_LO	16	R	0000h	49.5.38/ 4405
100A	PCIE_PHY_TX_ASIC_DRV_HI	16	R	0000h	49.5.39/ 4406
100B	PCIE_PHY_TX_ASIC_OUT	16	R	0000h	49.5.40/ 4406
100C	PCIE_PHY_RX_ASIC_IN	16	R	0000h	49.5.41/ 4407

Table continues on the next page...

PCIE_PHY memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
100D	PCIE_PHY_RX_ASIC_OUT	16	R	0000h	49.5.42/ 4408
1011	PCIE_PHY_TX_VMD_FSM_TX_VCM_0	16	R	0000h	49.5.43/ 4409
1012	PCIE_PHY_TX_VMD_FSM_TX_VCM_1	16	R	0000h	49.5.44/ 4409
1013	PCIE_PHY_TX_VMD_FSM_TX_VCM_DEBUG_IN	16	R/W	0000h	49.5.45/ 4410
1014	PCIE_PHY_TX_VMD_FSM_TX_VCM_DEBUG_OUT	16	R	0000h	49.5.46/ 4411
1015	PCIE_PHY_TX_LBERT_CTL	16	R/W	0000h	49.5.47/ 4411
1016	PCIE_PHY_RX_LBERT_CTL	16	R/W	0000h	49.5.48/ 4412
1017	PCIE_PHY_RX_LBERT_ERR	16	R/W	0000h	49.5.49/ 4413
1018	PCIE_PHY_RX_SCOPE_CTL	16	R/W	0000h	49.5.50/ 4413
1019	PCIE_PHY_RX_SCOPE_PHASE	16	R/W	0000h	49.5.51/ 4414
101A	PCIE_PHY_RX_DPLL_FREQ	16	R/W	0000h	49.5.52/ 4414
101B	PCIE_PHY_RX_CDR_CTL	16	R/W	000Fh	49.5.53/ 4415
101C	PCIE_PHY_RX_CDR_CDR_FSM_DEBUG	16	R	0000h	49.5.54/ 4416
101D	PCIE_PHY_RX_CDR_LOCK_VEC_OVRD	16	R/W	8000h	49.5.55/ 4417
101E	PCIE_PHY_RX_CDR_LOCK_VEC	16	R	0000h	49.5.56/ 4418
101F	PCIE_PHY_RX_CDR_ADAP_FSM	16	R	0000h	49.5.57/ 4418
1020	PCIE_PHY_RX_ATB0	16	R/W	0000h	49.5.58/ 4419
1021	PCIE_PHY_RX_ATB1	16	R/W	0000h	49.5.59/ 4420
1022	PCIE_PHY_RX_ENPWR0	16	R/W	0000h	49.5.60/ 4420
1023	PCIE_PHY_RX_PMIX_PHASE	16	R/W	0000h	49.5.61/ 4421
1024	PCIE_PHY_RX_ENPWR1	16	R/W	0000h	49.5.62/ 4422
1025	PCIE_PHY_RX_ENPWR2	16	R/W	0000h	49.5.63/ 4423

Table continues on the next page...

PCIE_PHY memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1026	PCIE_PHY_RX_SCOPE	16	R/W	0000h	49.5.64/ 4424
102B	PCIE_PHY_TX_TXDRV_CNTRL	16	R/W	0000h	49.5.65/ 4425
102C	PCIE_PHY_TX_POWER_CTL	16	R/W	0000h	49.5.66/ 4426
102D	PCIE_PHY_TX_ALT_BLOCK	16	R/W	0000h	49.5.67/ 4427
102E	PCIE_PHY_TX_ALT_AND_LOOPBACK	16	R/W	0000h	49.5.68/ 4428
102F	PCIE_PHY_TX_TX_ATB_REG	16	R/W	0000h	49.5.69/ 4429

49.5.1 Register ID Low 16 bits (PCIE_PHY_IDCODE_LO)

Address: 0h base + 0h offset = 0h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	IDCODE_LO															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_PHY_IDCODE_LO field descriptions

Field	Description
IDCODE_LO	Data

49.5.2 Register ID High 16 bits (PCIE_PHY_IDCODE_HI)

Address: 0h base + 1h offset = 1h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	IDCODE_HI															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_PHY_IDCODE_HI field descriptions

Field	Description
IDCODE_HI	Data

49.5.3 Debug Register (PCIE_PHY_DEBUG)

Address: 0h base + 2h offset = 2h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	Reserved										DTB_SEL		TX_VREF_SEL			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0

PCIE_PHY_DEBUG field descriptions

Field	Description
15–6 -	This field is reserved. Reserved
6–5 DTB_SEL	Description: The lane DTB's are ORed together with the support DTB signals selected with the following encodings. 00 None 01 reset_ctl DTB output 10 Scope DTB output 11 rtune DTB output
TX_VREF_SEL	-

49.5.4 Debug Register (PCIE_PHY_RTUNE_DEBUG)

Address: 0h base + 3h offset = 3h

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved	VALUE						
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	VALUE			TYPE		SET_VAL	MAN_TUNE	FLIP_COMP
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_RTUNE_DEBUG field descriptions

Field	Description
15 -	This field is reserved. Reserved
14–5 VALUE	Value to use when triggering SET_VAL field. Only the 6 LSB's are used when setting Rx cal or Tx cal values.
4–3 TYPE	Type of manual tuning or register read/write to execute. 00 ADC, or read/write rt_value 01 Rx tune, or read/write rx_cal_val (only 6 bits)

Table continues on the next page...

PCIE_PHY_RTUNE_DEBUG field descriptions (continued)

Field	Description
	10 Tx tune, or read/write tx_cal_val (only 6 bits) 11 Resref detect (no affect when triggering SET_VAL fi
2 SET_VAL	Sets value. Write a 1 to manually write the register specified by the TYPE field to the value in the VALUE field.
1 MAN_TUNE	Write a 1 to perform a manual tuning specified by the TYPE field. Starting a manual tune while a tune is currently running can cause unpredictable results. For use only when you know what the part is doing (with respect to resistor tuning). **NOTE:** Write a 1 to perform an operation. Subsequent writes with the bit set will trigger the operation. No need to clear (0) the bit between writes.
0 FLIP_COMP	Inverts Analog Comparator Output.

49.5.5 PCIE_PHY_RTUNE_STAT

Address: 0h base + 4h offset = 4h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	Reserved								STAT							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_PHY_RTUNE_STAT field descriptions

Field	Description
15–10 -	This field is reserved. Reserved
STAT	Current value of the register specifed by the RTUNE_DEBUG[TYPE] field.

49.5.6 PCIE_PHY_SS_PHASE

Address: 0h base + 5h offset = 5h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	Reserved								DTHR							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_PHY_SS_PHASE field descriptions

Field	Description
15–10 -	This field is reserved. Reserved
DTHR	Current value of the register specifed by the RTUNE_DEBUG[TYPE] field.

49.5.7 PCIE_PHY_SS_FREQ

Address: 0h base + 6h offset = 6h

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved	FREQ_ OVRD	FREQ_PK					
Reset	0	0	1	1	0	0	1	1

Bit	7	6	5	4	3	2	1	0
Read Write	FREQ_PK	FREQ_CNT_INIT						
Reset	0	0	1	0	0	1	1	1

PCIE_PHY_SS_FREQ field descriptions

Field	Description
15 -	This field is reserved. Reserved
14 FREQ_OVRD	Frequency register override. Spread spectrum clocking must be enabled to read from or write to this register. **NOTE:** Must be set for PHASE writes to stick.
13–7 FREQ_PK	Peak frequency value (for changing direction). Spread spectrum clocking must be enabled to read from or write to this register.
FREQ_CNT_INIT	Initial frequency counter value. Spread spectrum clocking must be enabled to read from or write to this register.

49.5.8 PCIE_PHY_ATEOVRD

Address: 0h base + 10h offset = 10h

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	Reserved				ateovrd_en	ref_usb2_en	ref_clkdiv2	Reserved
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_ATEOVRD field descriptions

Field	Description
15–4 -	This field is reserved. Reserved
3 ateovrd_en	Override enable for ATE signals.

Table continues on the next page...

PCIE_PHY_ATEOVRD field descriptions (continued)

Field	Description
2 ref_usb2_en	Override value for HSPHY ref_clk enable.
1 ref_clkdiv2	Override value for SSP ref_clk prescaler.
0 -	This field is reserved. Reserved

49.5.9 PCIE_PHY_MPLL_OVRD_IN_LO

Address: 0h base + 11h offset = 11h

Bit	15	14	13	12	11	10	9	8
Read Write	RES_ACK_IN_OVRD	RES_ACK_IN	RES_REQ_IN_OVRD	RES_REQ_IN	RTUNE_REQ_OVRD	RTUNE_REQ	MPLL_MULTIPLIER_OVRD	MPLL_MULTIPLIER
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	MPLL_MULTIPLIER						MPLL_EN_OVRD	MPLL_EN
Reset	0	1	0	0	1	1	0	0

PCIE_PHY_MPLL_OVRD_IN_LO field descriptions

Field	Description
15 RES_ACK_IN_OVRD	Override enable for res_ack_in.
14 RES_ACK_IN	Override value for res_ack_in.
13 RES_REQ_IN_OVRD	Override enable for res_req_in.
12 RES_REQ_IN	Override value for res_req_in.
11 RTUNE_REQ_OVRD	Override enable for rtune_req.
10 RTUNE_REQ	Override value for rtune_req.
9 MPLL_MULTIPLIER_OVRD	Override enable for mpll_multiplier.

Table continues on the next page...

PCIE_PHY_MPLL_OVRD_IN_LO field descriptions (continued)

Field	Description
8–2 MPLL_ MULTIPLIER	Override value for mpll_multiplier.
1 MPLL_EN_ OVRD	Override enable for mpll_en.
0 MPLL_EN	Override value for mpll_en.

49.5.10 PCIE_PHY_MPLL_OVRD_IN_HI

Address: 0h base + 11h offset = 11h

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved					MPLL_RST	FSEL_OVR	FSEL
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	FSEL		MPLL_ WORD_ CLK_EN_ OVRD	MPLL_ WORD_ CLK_EN	MPLL_ DWORD_ CLK_EN_ OVRD	MPLL_ DWORD_ CLK_EN	MPLL_ QWORD_ CLK_EN_ OVRD	MPLL_ QWORD_ CLK_EN
Reset	0	1	0	0	1	1	0	0

PCIE_PHY_MPLL_OVRD_IN_HI field descriptions

Field	Description
15–11 -	This field is reserved. Reserved.
10 MPLL_RST	Resets the MPLL state machine. Writing the register with this bit set will reset the MPLL power-up/down FSM, regardless of the current state of the register bit.
9 FSEL_OVR	Override enable for fsel[2:0].
8–6 FSEL	: Override value for fsel[2:0].
5 MPLL_WORD_ CLK_EN_OVRD	Override enable for mpll_word_clk_en.
4 MPLL_WORD_ CLK_EN	Override value for mpll_word_clk_en.
3 MPLL_DWORD_ CLK_EN_OVRD	Override enable for mpll_dword_clk_en.

Table continues on the next page...

PCIE_PHY_MPLL_OVRD_IN_HI field descriptions (continued)

Field	Description
2 MPLL_DWORD_CLK_EN	Override value for mpll_dword_clk_en.
1 MPLL_QWORD_CLK_EN_OVRD	Override enable for mpll_qword_clk_en.
0 MPLL_QWORD_CLK_EN	Override value for mpll_qword_clk_en.

49.5.11 PCIE_PHY_SSC_OVRD_IN

Address: 0h base + 13h offset = 13h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	Reserved				SSC_OVRD_IN_EN	SSC_EN	SSC_RANGE		SSC_REF_CLK_SEL							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_PHY_SSC_OVRD_IN field descriptions

Field	Description
15–12 -	This field is reserved. Reserved
11 SSC_OVRD_IN_EN	Override enable for Spread Spectrum generator.
10 SSC_EN	Override value for SSC enable.
9–8 SSC_RANGE	Override value for SSC modulation range.
SSC_REF_CLK_SEL	Override value for reference clock scaling.

49.5.12 PCIE_PHY_BS_OVRD_IN

Address: 0h base + 14h offset = 14h

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved				EN	INVERT	INIT	HIGHZ
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	CLAMP	EXTEST_ AC	EXTEST	PRELOAD	UPDATE_ DR	CAPTURE_ DR	SHIFT_DR	IN
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_BS_OVRD_IN field descriptions

Field	Description
15–12 -	This field is reserved. Reserved.
11 EN	Enables override values for all inputs controlled by this register.
10 INVERT	Override value for bs_invert.
9 INIT	Override value for bs_init.
8 HIGHZ	Override value for bs_highz.
7 CLAMP	Override value for bs_clamp.
6 EXTEST_AC	Override value for bs_extest_ac.
5 EXTEST	Override value for bs_extest.
4 PRELOAD	Override value for bs_preload.
3 UPDATE_DR	Override value for bs_update_dr.
2 CAPTURE_DR	Override value for bs_capture_dr
1 SHIFT_DR	Override value for bs_shift_dr.
0 IN	Override value for bs_shift_dr.

49.5.13 PCIE_PHY_LEVEL_OVRD_IN

Address: 0h base + 15h offset = 15h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read / Write	Reserved					EN	ACJT_LEVEL					LOS_LEVEL				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_PHY_LEVEL_OVRD_IN field descriptions

Field	Description
15–11 -	This field is reserved. Reserved.
10 EN	Enables override values for all inputs controlled by this register.
9–5 ACJT_LEVEL	Override value for acjt_level.
LOS_LEVEL	Override value for los_level.

49.5.14 PCIE_PHY_SUP_OVRD_OUT

Address: 0h base + 16h offset = 16h

Bit	15	14	13	12	11	10	9	8
Read / Write	Reserved						MPLL_STATE_OVRD	MPLL_STATE
Reset	0	0	0	0	0	0	0	1

Bit	7	6	5	4	3	2	1	0
Read / Write	BS_OUT_OVRD	BS_OUT	RTUNE_ACK_OVRD	RTUNE_ACK	RES_REQ_OUT_OVRD	RES_REQ_OUT	RES_ACK_OUT_OVRD	RES_ACK_OUT
Reset	0	0	0	0	0	0	0	1

PCIE_PHY_SUP_OVRD_OUT field descriptions

Field	Description
15–10 -	This field is reserved. Reserved.
9 MPLL_STATE_OVRD	Override enable for mpll_state output.
8 MPLL_STATE	Override value for mpll_state output.
7 BS_OUT_OVRD	Override enable for bs_out output.

Table continues on the next page...

PCIE_PHY_SUP_OVRD_OUT field descriptions (continued)

Field	Description
6 BS_OUT	Override value for bs_out output.
5 RTUNE_ACK_ OVRD	Override enable for rtune_ack output.
4 RTUNE_ACK	Override value for rtune_ack output.
3 RES_REQ_ OUT_OVRD	Override enable for res_req_out output.
2 RES_REQ_OUT	Override value for res_req_out output.
1 RES_ACK_OUT_ OVRD	Override enable for res_ack_out output.
0 RES_ACK_OUT	Override value for res_ack_out output.

49.5.15 PCIE_PHY_MPLL_ASIC_IN

Address: 0h base + 17h offset = 17h

Bit	15	14	13	12	11	10	9	8
Read	Reserved		MPLL_ WORD_ CLK_EN	MPLL_ DWORD_ CLK_EN	MPLL_ QWORD_ CLK_EN	RES_ACK_ IN	RES_REQ_ IN	RTUNE_ REQ
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	MPLL_MULTIPLIER							MPLL_EN
Write								
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_MPLL_ASIC_IN field descriptions

Field	Description
15–14 -	This field is reserved. Reserved.
13 MPLL_WORD_ CLK_EN	Value from ASIC for mpll_word_clk_en.

Table continues on the next page...

PCIE_PHY_MPLL_ASIC_IN field descriptions (continued)

Field	Description
12 MPLL_DWORD_CLK_EN	Value from ASIC for mpll_dword_clk_en.
11 MPLL_QWORD_CLK_EN	Value from ASIC for mpll_qword_clk_en.
10 RES_ACK_IN	Value from ASIC for res_ack_in.
9 RES_REQ_IN	Value from ASIC for res_req_in.
8 RTUNE_REQ	Value from ASIC for rtune_req.
7–1 MPLL_MULTIPLIER	Value from ASIC for mpll_multiplier.
0 MPLL_EN	Value from ASIC for mpll_en.

49.5.16 PCIE_PHY_BS_ASIC_IN

Address: 0h base + 18h offset = 18h

Bit	15	14	13	12	11	10	9	8
Read	Reserved					INVERT	INIT	HIGHZ
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	CLAMP	EXTEST_AC	EXTEST	PRELOAD	UPDATE_DR	CAPTURE_DR	SHIFT_DR	IN
Write								
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_BS_ASIC_IN field descriptions

Field	Description
15–11 -	This field is reserved. Reserved.
10 INVERT	Value from ASIC for bs_invert.
9 INIT	Value from ASIC for bs_init.
8 HIGHZ	Value from ASIC for bs_highz.

Table continues on the next page...

PCIE_PHY_BS_ASIC_IN field descriptions (continued)

Field	Description
7 CLAMP	Value from ASIC for bs_clamp.
6 EXTEST_AC	Value from ASIC for bs_extest_ac.
5 EXTEST	Value from ASIC for bs_extest.
4 PRELOAD	Value from ASIC for bs_preload.
3 UPDATE_DR	Value from ASIC for bs_update_dr.
2 CAPTURE_DR	Value from ASIC for bs_capture_dr.
1 SHIFT_DR	Value from ASIC for bs_shift_dr.
0 IN	Value from ASIC for bs_in.

49.5.17 PCIE_PHY_LEVEL_ASIC_IN

Address: 0h base + 19h offset = 19h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	Reserved						ACJT_LEVEL					LOS_LEVEL				
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_PHY_LEVEL_ASIC_IN field descriptions

Field	Description
15–10 -	This field is reserved. Reserved.
9–5 ACJT_LEVEL	Value from ASIC for acjt_level.
LOS_LEVEL	Value from ASIC for los_level.

49.5.18 PCIE_PHY_SSC_ASIC_IN

Address: 0h base + 1Ah offset = 1Ah

Bit	15	14	13	12	11	10	9	8
Read	Reserved		SS_EN	SSC_RANGE		SSC_REF_CLK_SEL		
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	SSC_REF_CLK_SEL						FSEL	
Write								
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_SSC_ASIC_IN field descriptions

Field	Description
15–14 -	This field is reserved. Reserved.
13 SS_EN	Value from ASIC for ssc_en.
12–11 SSC_RANGE	Value from ASIC for ssc_range.
10–3 SSC_REF_CLK_SEL	Value from ASIC for ssc_ref_clk_sel
FSEL	Value from ASIC for fsel.

49.5.19 PCIE_PHY_SUP_ASIC_OUT

Address: 0h base + 1Bh offset = 1Bh

Bit	15	14	13	12	11	10	9	8
Read	Reserved							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	Reserved			MPLL_STATE	BS_OUT	RTUNE_ACK	RES_REQ_OUT	RES_ACK_OUT
Write								
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_SUP_ASIC_OUT field descriptions

Field	Description
15–5 -	This field is reserved. Reserved.
4 MPLL_STATE	Value from PHY for mpll_state output.
3 BS_OUT	Value from PHY for bs_out output.
2 RTUNE_ACK	Value from PHY for rtune_ack output.
1 RES_REQ_OUT	Value from PHY for res_req_out output.
0 RES_ACK_OUT	Value from PHY for res_ack_out output.

49.5.20 PCIE_PHY_ATEOVRD_STATUS

Address: 0h base + 1Ch offset = 1Ch

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	Reserved							REF_SSP_EN	REF_USE_PAD	PHY_RESET_IN	REF_CLKDIV2_IN	REF_USB2_EN_IN	ATEOVRD_EN	PHY_RESET_OUT	REF_CLKDIV2_OUT	REF_USB2_EN_OUT
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_PHY_ATEOVRD_STATUS field descriptions

Field	Description
15–9 -	This field is reserved. Reserved.
8 REF_SSP_EN	Value from ASIC for ref_ssp_en.
7 REF_USE_PAD	Value from ASIC for ref_use_pad
6 PHY_RESET_IN	Value from ASIC for phy_reset
5 REF_CLKDIV2_IN	Value from ASIC for ref_clkdiv2.
4 REF_USB2_EN_IN	Value from ASIC for ref_usb2_en.
3 ATEOVRD_EN	When set, values from ATEOVRD register are sent to PHY.
2 PHY_RESET_OUT	Value from ATEOVRD for phy_reset.
1 REF_CLKDIV2_OUT	Value from ATEOVRD for ref_clkdiv2.
0 REF_USB2_EN_OUT	Value from ATEOVRD for ref_usb2_en.

49.5.21 PCIE_PHY_SCOPE_ENABLES

Address: 0h base + 20h offset = 20h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	Reserved													MASK_SATURATION_MODE	MASK_EN	XOR_EN
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_PHY_SCOPE_ENABLES field descriptions

Field	Description
15–3 -	This field is reserved. Reserved.

Table continues on the next page...

PCIE_PHY_SCOPE_ENABLES field descriptions (continued)

Field	Description
2 MASK_ SATURATION_ MODE	Method of mask saturation. 1 Saturates when the first mask_counter reaches sample_limit. 0 Saturates when all mask_counters have reached sample_limit.
1 MASK_EN	Enables scope_mask input for tracking count values. Clears registers when deasserted.
0 XOR_EN	Uses scope_xor input for count values.

49.5.22 PCIE_PHY_SCOPE_SAMPLES

Address: 0h base + 21h offset = 21h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	SAMPLES															
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

PCIE_PHY_SCOPE_SAMPLES field descriptions

Field	Description
SAMPLES	Number of samples to count.

49.5.23 PCIE_PHY_SCOPE_COUNT

Address: 0h base + 22h offset = 22h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	COUNT															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

PCIE_PHY_SCOPE_COUNT field descriptions

Field	Description
COUNT	A write to this register starts the counting process. The value of FFFF indicates counting still in progress. If in MASK mode, asserting MASK_EN also starts the counting

49.5.24 PCIE_PHY_SCOPE_CTL

Address: 0h base + 23h offset = 23h

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	Reserved						COUNT	MASK_ SATURATI ON
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_SCOPE_CTL field descriptions

Field	Description
15–2 -	This field is reserved. Reserved.
1 COUNT	A write to this register starts the counting process. The value of FFFF indicates counting still in progress. If in MASK mode, asserting MASK_EN also starts the counting
0 MASK_ SATURATION	When asserted, mask registers have saturated.

49.5.25 PCIE_PHY_SCOPE_MASK*n*

Address: 0h base + 24h offset + (1d × i), where i=0d to 7d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	MASK_VAL_n															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_PHY_SCOPE_MASK*n* field descriptions

Field	Description
MASK_VAL_n	Starting count value of mask register. Scope must be enabled to read from or write to this register.

49.5.26 PCIE_PHY_MPLL_LOOP_CTL

Address: 0h base + 30h offset = 30h

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	PROP_CNTRL				INT_CNTRL		VBF_SF	VMB
Reset	1	1	0	0	0	0	0	0

PCIE_PHY_MPLL_LOOP_CTL field descriptions

Field	Description
15–8 -	This field is reserved. Reserved.
7–4 PROP_CNTRL	Charge pump proportional current setting.
3–2 INT_CNTRL	Charge pump integrating current setting.
1 VBF_SF	Measures MPLL VBF_SF (RC filtered gate voltage for VPSF source follower).
0 VMB	Measures MPLL master bias voltage.

49.5.27 PCIE_PHY_MPLL_ATB_MEAS2

Address: 0h base + 32h offset = 32h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	Reserved								IVCO_FILT	VCNTRL_M	VCNTRL_P	ATB_SENSE_SEL	MEAS_TEMP	FRC_PMIX_VPMIX	EN_MPMIX_VPMIX	EN_MPMIX_TST
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_PHY_MPLL_ATB_MEAS2 field descriptions

Field	Description
15–8 -	This field is reserved. Reserved.
7 IVCO_FILT	Puts filtered version of ivco on atb_s_p

Table continues on the next page...

PCIE_PHY_MPLL_ATB_MEAS2 field descriptions (continued)

Field	Description
6 VCNTRL_M	Puts dcc output vcntrl_p on atb_s_m
5 VCNTRL_P	Puts dcc output vcntrl_m on atb_s_p
4 ATB_SENSE_SEL	connects internal atb sense bus to external bus
3 MEAS_TEMP	Instructs POR block to measure the temperature.
2 FRC_PMIX_VPMIX	Forces mpll_pmix_vreg to use atb_s_m as its input instead of vbg.
1 EN_MPMIX_VPMIX	Puts vreg_pmix on atb_s_p.
0 EN_MPMIX_TST	Enables XOR gate to test linearity of MPLL phase mixer.

49.5.28 PCIE_PHY_MPLL_OVR

Address: 0h base + 33h offset = 33h

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	PWRON_LCL	EN_PWRON_LCL	GS_LCL	EN_GS_LCL	RST_LCL	EN_RST_LCL	PMIX_CLK_SEL_LCL	EN_PMIX_CLK_SEL_LCL
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_MPLL_OVR field descriptions

Field	Description
15–8 -	This field is reserved. Reserved.

Table continues on the next page...

PCIE_PHY_MPLL_OVR field descriptions (continued)

Field	Description
7 PWRON_LCL	local power_on value
6 EN_PWRON_LCL	Enables local control of power_on
5 GS_LCL	local gear_shift value
4 EN_GS_LCL	Enables local control of gear_shift
3 RST_LCL	local Reset value
2 EN_RST_LCL	enable local control of reset
1 PMIX_CLK_SEL_LCL	local pmix_clk_sel value
0 EN_PMIX_CLK_SEL_LCL	enable local control of pmix_clk_sel

49.5.29 PCIE_PHY_RTUNE_RTUNE_CTRL

Address: 0h base + 34h offset = 34h

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	RT_PWroN_FRC_ON	X4_FRC_OFF	RT_DAC_MODE		RT_DAC_CHOP	RT_ATB	RT_SEL_ATBP	RT_SEL_ATBF
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_RTUNE_RTUNE_CTRL field descriptions

Field	Description
15–8 -	This field is reserved. Reserved.
7 RT_PWroN_FRC_ON	When set, forces RTUNE block to be on
6 X4_FRC_OFF	When set, do not multiply test current by 4
5–4 RT_DAC_MODE	Margin DAC mode control bits

Table continues on the next page...

PCIE_PHY_RTUNE_RTUNE_CTRL field descriptions (continued)

Field	Description
	00 powerdown 01 DAC drives atb_s_p/m directly 10 DAC drives atb_s_p/m to the RX in margining mode 11 illegal state
3 RT_DAC_CHOP	Margin DAC chop control bit
2 RT_ATB	RTUNE ATB mode control bit 1 RTUNE performs ADC on ATB input 0 not accessing ATB
1 RT_SEL_ATBP	RTUNE ATB sense input select bit 1 atb_s_p 0 atb_s_m
0 RT_SEL_ATBF	RTUNE ATB input slect bit 1 atb_fm 0 atb_s_p/m

49.5.30 PCIE_PHY_TX_OVRD_IN_LO

Address: 0h base + 1000h offset = 1000h

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved		TX_DETECT_RX_REQ_OVRD	TX_DETECT_RX_REQ	TX_BEACON_EN_OVRD	TX_BEACON_EN	TX_CM_EN_OVRD	TX_CM_EN
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	TX_EN_OVRD	TX_EN	TX_DATA_EN_OVRD	TX_DATA_EN	TX_INVERT_OVRD	TX_INVERT	TX_LOOPBK_EN_OVRD	LOOPBK_EN
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_TX_OVRD_IN_LO field descriptions

Field	Description
15–14 -	This field is reserved. Reserved.

Table continues on the next page...

PCIE_PHY_TX_OVRD_IN_LO field descriptions (continued)

Field	Description
13 TX_DETECT_ RX_REQ_OVRD	Override enable for tx_detect_rx_req
12 TX_DETECT_ RX_REQ	Override value for tx_detect_rx_req
11 TX_BEACON_ EN_OVRD	Override enable for tx_beacon_en
10 TX_BEACON_ EN	Override value for tx_beacon_en
9 TX_CM_EN_ OVRD	Override enable for tx_cm_en
8 TX_CM_EN	Override value for tx_cm_en
7 TX_EN_OVRD	Override enable for tx_en
6 TX_EN	Override value for tx_en
5 TX_DATA_EN_ OVRD	Override enable for tx_data_en
4 TX_DATA_EN	Override value for tx_data_en
3 TX_INVERT_ OVRD	Override enable for tx_invert
2 TX_INVERT	Override value for tx_invert
1 TX_LOOPBK_ EN_OVRD	Override enable for loopbk_en
0 LOOPBK_EN	Override value for loopbk_en

49.5.31 PCIE_PHY_TX_OVRD_IN_HI

Address: 0h base + 1001h offset = 1001h

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	TX_ RESET_ OVRD	TX_RESET	TX_ NYQUIST_ DATA	TX_CLK_ OUT_EN_ OVRD	TX_CLK_ OUT_EN	TX_RATE_ OVRD	TX_RATE	
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_TX_OVRD_IN_HI field descriptions

Field	Description
15–8 -	This field is reserved. Reserved.
7 TX_RESET_ OVRD	Override enable for tx_reset
6 TX_RESET	Override value for tx_reset
5 TX_NYQUIST_ DATA	Override incoming data to nyquist
4 TX_CLK_OUT_ EN_OVRD	Override enable for tx_clk_out_en.
3 TX_CLK_OUT_ EN	Override incomming tx_clk_out_en.
2 TX_RATE_ OVRD	Override enable for tx_rate.
TX_RATE	Override incomming tx lane rate.

49.5.32 PCIE_PHY_TX_OVRD_DRV_LO

Address: 0h base + 1003h offset = 1003h

Bit	15	14	13	12	11	10	9	8
Read / Write	Reserved	EN	PREEMPH					
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read / Write	PREEMPH	AMPLITUDE						
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_TX_OVRD_DRV_LO field descriptions

Field	Description
15 -	This field is reserved. Reserved.
14 EN	Enables override values for all inputs controlled by this register
13–7 PREEMPH	Override value for transmit preemphasis
AMPLITUDE	Override value for transmit amplitude.

49.5.33 PCIE_PHY_TX_OVRD_OUT

Address: 0h base + 1004h offset = 1004h

Bit	15	14	13	12	11	10	9	8
Read / Write	Reserved							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	TX_STATE_OVRD	TX_STATE	TX_CM_STATE_OVRD	TX_CM_STATE	TX_DETECT_RX_ACK_OVRD	TX_DETECT_RX_ACK	DETECT_RX_RES_OVRD	DETECT_RX_RES
Write								
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_TX_OVRD_OUT field descriptions

Field	Description
15–8 -	This field is reserved. Reserved.
7 TX_STATE_OVRD	Override enable for tx_state

Table continues on the next page...

PCIE_PHY_TX_OVRD_OUT field descriptions (continued)

Field	Description
6 TX_STATE	Override value for tx_state
5 TX_CM_STATE_OVRD	Override enable for tx_cm_state
4 TX_CM_STATE	Override value for tx_cm_state
3 TX_DETECT_RX_ACK_OVRD	Override enable for tx_detect_rx_ack
2 TX_DETECT_RX_ACK	Override value for tx_detect_rx_ack
1 DETECT_RX_RES_OVRD	Override enable for tx_detect_rx_res
0 DETECT_RX_RES	Override value for tx_detect_rx_res

49.5.34 PCIE_PHY_RX_OVRD_IN_LO

Address: 0h base + 1005h offset = 1005h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	Reserved		RX_LOS_EN_OVRD	RX_LOS_EN	RX_TERM_EN_OVRD	RX_TERM_EN	RX_BIT_SHIFT_OVRD	RX_BIT_SHIFT	RX_ALIGN_EN_OVRD	RX_ALIGN_EN	RX_DATA_EN_OVRD	RX_DATA_EN	RX_PLL_EN_OVRD	RX_PLL_EN	RX_INVERT_OVRD	RX_INVERT
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_PHY_RX_OVRD_IN_LO field descriptions

Field	Description
15–14 -	This field is reserved. Reserved.
13 RX_LOS_EN_OVRD	Override enable for rx_los_en
12 RX_LOS_EN	Override value for rx_los_en
11 RX_TERM_EN_OVRD	Override enable for rx_term_en

Table continues on the next page...

PCIE_PHY_RX_OVRD_IN_LO field descriptions (continued)

Field	Description
10 RX_TERM_EN	Override value for rx_term_en
9 RX_BIT_SHIFT_OVRD	Override enable for rx_bit_shift
8 RX_BIT_SHIFT	Override value for rx_bit_shift
7 RX_ALIGN_EN_OVRD	Override enable for rx_align_en
6 RX_ALIGN_EN	Override value for rx_align_en
5 RX_DATA_EN_OVRD	Override enable for rx_data_en
4 RX_DATA_EN	Override value for rx_data_en
3 RX_PLL_EN_OVRD	Override enable for rx_pll_en
2 RX_PLL_EN	Override value for rx_pll_en
1 RX_INVERT_OVRD	Override enable for rx_invert
0 RX_INVERT	Override value for rx_invert

49.5.35 PCIE_PHY_RX_OVRD_IN_HI

Address: 0h base + 1006h offset = 1006h

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved		RX_RESET_OVRD	RX_RESET	RX_EQ_OVRD	RX_EQ		
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	RX_EQ_EN_OVRD	RX_EQ_EN	RX_LOS_FILTER_OVRD	RX_LOS_FILTER		RX_RATE_OVRD	RX_RATE	
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_RX_OVRD_IN_HI field descriptions

Field	Description
15–14 -	This field is reserved. Reserved.
13 RX_RESET_ OVRD	Override enable for rx_reset
12 RX_RESET	Override value for rx_reset
11 RX_EQ_OVRD	Override enable for rx_eq
10–8 RX_EQ	Override value for rx_eq
7 RX_EQ_EN_ OVRD	Override enable for rx_eq_en
6 RX_EQ_EN	Override value for rx_eq_en
5 RX_LOS_ FILTER_OVRD	Override enable for rx_los_filter
4–3 RX_LOS_ FILTER	Override value for rx_los_filter
2 RX_RATE_ OVRD	Override enable for rx_rate
RX_RATE	Override value for rx_rate

49.5.36 PCIE_PHY_RX_OVRD_OUT

Address: 0h base + 1007h offset = 1007h

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	Reserved	ZERO_ DATA	LOS_OVRD	LOS	PLL_ STATE_ OVRD	PLL_STATE	VALID_ OVRD	VALID
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_RX_OVRD_OUT field descriptions

Field	Description
15–7 -	This field is reserved. Reserved.
6 ZERO_DATA	Override data output to all zeros
5 LOS_OVRD	Override value for rx_los
4 LOS	Override value for rx_los
3 PLL_STATE_ OVRD	Override enable for rx_pll_state
2 PLL_STATE	Override value for rx_pll_state
1 VALID_OVRD	Override enable for rx_valid
0 VALID	Override value for rx_valid

49.5.37 PCIE_PHY_TX_ASIC_IN

Address: 0h base + 1008h offset = 1008h

Bit	15	14	13	12	11	10	9	8
Read	Reserved					TX_CLK_ OUT_EN	DETECT_ RX_REQ	BEACON_ EN
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	CM_EN	TX_EN	DATA_EN	TX_RESET	INVERT	LOOPBK_ EN	TX_RATE	
Write								
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_TX_ASIC_IN field descriptions

Field	Description
15–11 -	This field is reserved. Reserved.
10 TX_CLK_OUT_ EN	Value from ASIC for tx_clk_out_en

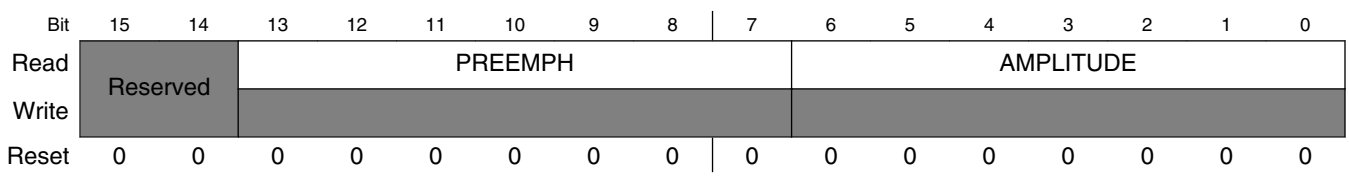
Table continues on the next page...

PCIE_PHY_TX_ASIC_IN field descriptions (continued)

Field	Description
9 DETECT_RX_ REQ	Value from ASIC for tx_detect_rx_req
8 BEACON_EN	Value from ASIC for tx_beacon_en
7 CM_EN	Value from ASIC for tx_cm_en
6 TX_EN	Value from ASIC for tx_en
5 DATA_EN	Value from ASIC for tx_data_en
4 TX_RESET	Value from ASIC for tx_reset
3 INVERT	Value from ASIC for tx_invert
2 LOOPBK_EN	Value from ASIC for loopbk_en
TX_RATE	Value from ASIC for tx_rate

49.5.38 PCIE_PHY_TX_ASIC_DRV_LO

Address: 0h base + 1009h offset = 1009h



PCIE_PHY_TX_ASIC_DRV_LO field descriptions

Field	Description
15–14 -	This field is reserved. Reserved.
13–7 PREEMPH	Value from ASIC for tx_preemph
AMPLITUDE	Value from ASIC for tx_amplitude

49.5.39 PCIE_PHY_TX_ASIC_DRV_HI

Address: 0h base + 100Ah offset = 100Ah

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	Reserved											TERM_OFFSET				
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_PHY_TX_ASIC_DRV_HI field descriptions

Field	Description
15–5 -	This field is reserved. Reserved.
TERM_OFFSET	Value from ASIC for tx_term_offset

49.5.40 PCIE_PHY_TX_ASIC_OUT

Address: 0h base + 100Bh offset = 100Bh

Bit	15	14	13	12	11	10	9	8
Read	Reserved							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	Reserved			STATE	CM_STATE	DETECT_ RX_ACK	0	DETECT_ RX_RES
Write								
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_TX_ASIC_OUT field descriptions

Field	Description
15–5 -	This field is reserved. Reserved.
4 STATE	Value from PHY for tx_state
3 CM_STATE	Value from PHY for tx_cm_state
2 DETECT_RX_ ACK	Value from PHY for tx_detect_rx_ack

Table continues on the next page...

PCIE_PHY_TX_ASIC_OUT field descriptions (continued)

Field	Description
1 Reserved	This read-only field is reserved and always has the value 0.
0 DETECT_RX_RES	Value from PHY for tx_detect_rx_res

49.5.41 PCIE_PHY_RX_ASIC_IN

Address: 0h base + 100Ch offset = 100Ch

Bit	15	14	13	12	11	10	9	8
Read	RX_EQ_EN	RX_EQ			LOS_FILTER		LOS_EN	TERM_EN
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	CLK_SHIFT	ALIGN_EN	DATA_EN	PLL_EN	RX_RESET	INVERT	RX_RATE	
Write								
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_RX_ASIC_IN field descriptions

Field	Description
15 RX_EQ_EN	Value from ASIC for rx_eq_en
14–12 RX_EQ	Value from ASIC for rx_eq
11–10 LOS_FILTER	Value from ASIC for rx_los_filter
9 LOS_EN	Value from ASIC for rx_los_en
8 TERM_EN	Value from ASIC for rx_term_en
7 CLK_SHIFT	Value from ASIC for rx_bit_shift
6 ALIGN_EN	Value from ASIC for rx_align_en
5 DATA_EN	Value from ASIC for rx_data_en
4 PLL_EN	Value from ASIC for rx_pll_en
3 RX_RESET	Value from ASIC for rx_reset

Table continues on the next page...

PCIE_PHY_RX_ASIC_IN field descriptions (continued)

Field	Description
2 INVERT	Value from ASIC for rx_invert
RX_RATE	Value from ASIC for rx_rate

49.5.42 PCIE_PHY_RX_ASIC_OUT

Address: 0h base + 100Dh offset = 100Dh

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	Reserved					LOS	PLL_STATE	VALID
Write								
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_RX_ASIC_OUT field descriptions

Field	Description
15–3 -	This field is reserved. Reserved.
2 LOS	Value from PHY for rx_los
1 PLL_STATE	Value from PHY for rx_pll_state
0 VALID	Value from PHY for rx_valid

49.5.43 PCIE_PHY_TX_VMD_FSM_TX_VCM_0

Address: 0h base + 1011h offset = 1011h

Bit	15	14	13	12	11	10	9	8
Read	Reserved	DONE	N_USE					
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	N_USE	N_TRISTATE						
Write								
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_TX_VMD_FSM_TX_VCM_0 field descriptions

Field	Description
15 -	This field is reserved. Reserved.
14 DONE	Configuration is done
13–7 N_USE	Value from VMD for legs to use
N_TRISTATE	Value from VMD for number of tristate legs.

49.5.44 PCIE_PHY_TX_VMD_FSM_TX_VCM_1

Address: 0h base + 1012h offset = 1012h

Bit	15	14	13	12	11	10	9	8
Read	FIXED_ DONE	TRA_DONE	N_FIXED					
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	N_FIXED	N_TRAILER						
Write								
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_TX_VMD_FSM_TX_VCM_1 field descriptions

Field	Description
15 FIXED_DONE	N_FIXED Multiplication has completed.
14 TRA_DONE	N_TRAILER Multiplication has completed.
13–7 N_FIXED	Value from VMD for number of fixed driver legs.
N_TRAILER	Value from VMD for number of trailer legs.

49.5.45 PCIE_PHY_TX_VMD_FSM_TX_VCM_DEBUG_IN

Address: 0h base + 1013h offset = 1013h

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved							
Reset	0	0	0	0	0	0	0	0

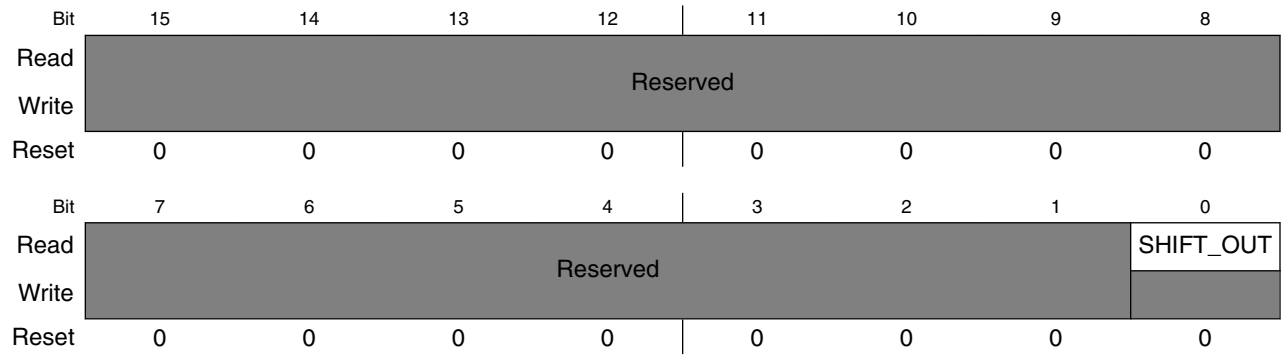
Bit	7	6	5	4	3	2	1	0
Read Write	Reserved				CONFIG_ OVRD	CONFIG_ LOAD	CONFIG_ CLK	CONFIG_ DATA
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_TX_VMD_FSM_TX_VCM_DEBUG_IN field descriptions

Field	Description
15–4 -	This field is reserved. Reserved
3 CONFIG_OVRD	Override the Voltage Mode Driver Configuration FSM and access the shift chain directly.
2 CONFIG_LOAD	Override value for the Voltage Mode Driver Configuration FSM's config load.
1 CONFIG_CLK	Override value for the Voltage Mode Driver Configuration FSM's config clk.
0 CONFIG_DATA	Override value for the Voltage Mode Driver Configuration FSM's config data.

49.5.46 PCIE_PHY_TX_VMD_FSM_TX_VCM_DEBUG_OUT

Address: 0h base + 1014h offset = 1014h

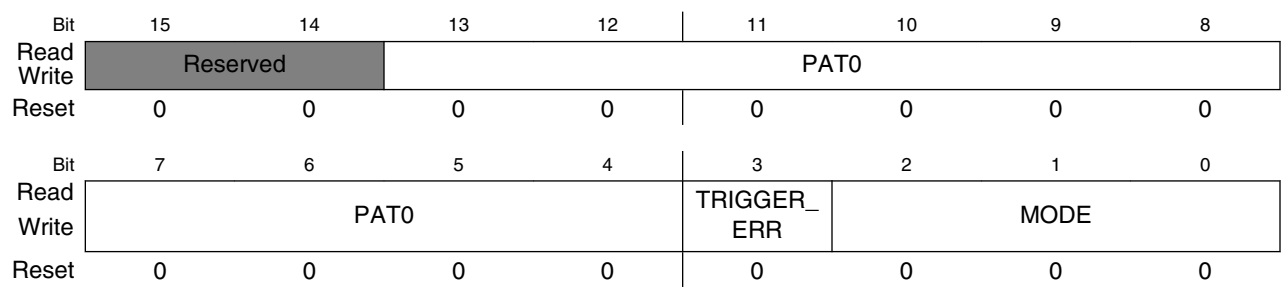


PCIE_PHY_TX_VMD_FSM_TX_VCM_DEBUG_OUT field descriptions

Field	Description
15–1 -	This field is reserved. Reserved
0 SHIFT_OUT	Current value from TX_ANAs configuration shift register.

49.5.47 PCIE_PHY_TX_LBERT_CTL

Address: 0h base + 1015h offset = 1015h



PCIE_PHY_TX_LBERT_CTL field descriptions

Field	Description
15–14 -	This field is reserved. Reserved
13–4 PAT0	Pattern for modes 3-5
3 TRIGGER_ERR	Insert a single error into a lsb Any write of a 1 to this bit will insert an error
MODE	Pattern to generate When changing modes, you must first change to disabled.

Table continues on the next page...

PCIE_PHY_TX_LBERT_CTL field descriptions (continued)

Field	Description
	6 DC-balanced word (PAT0) 5 Fixed word (PAT0) 4 lfsr7. X^7 + X^6 + 1 3 lfsr15. X^15 + X^14 + 1 2 lfsr23. X^23 + X^18 + 1 1 lfsr31. X^31 + X^28 + 1 0 Disabled

49.5.48 PCIE_PHY_RX_LBERT_CTL

Address: 0h base + 1016h offset = 1016h

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	Reserved				SYNC	MODE		
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_RX_LBERT_CTL field descriptions

Field	Description
15–4 -	This field is reserved. Reserved
3 SYNC	Synchronize pattern matcher LFSR with incoming data A write of a one to this bit will reset the error counter and start a synchronization of the PM. There is no need to write this back to zero to run normally.
MODE	Pattern to match When changing modes, you must first change to disabled. 7 d[n] = 6 d[n] = !d[n-10] 5 d[n] = d[n-10] 4 lfsr7 : X^7 + X^6 + 1 3 lfsr15: X^15 + X^14 + 1 2 lfsr23. X^23 + X^18 + 1 1 lfsr31. X^31 + X^28 + 1 0 Disabled

49.5.49 PCIE_PHY_RX_LBERT_ERR

Address: 0h base + 1017h offset = 1017h

Bit	15	14	13	12	11	10	9	8
Read Write	OV14	COUNT						
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	COUNT							
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_RX_LBERT_ERR field descriptions

Field	Description
15 OV14	If active, multiply COUNT by 128. If OV14=1 and COUNT=2^15-1, signals overflow of counter
COUNT	A read of this register, or a sync of the PM resets the error count. Current error count If OV14 field is active, then multiply count by 128

49.5.50 PCIE_PHY_RX_SCOPE_CTL

Address: 0h base + 1018h offset = 1018h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	Reserved													MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_PHY_RX_SCOPE_CTL field descriptions

Field	Description
15–3 -	This field is reserved. Reserved
MODE	Sampling mode of counters. **NOTE:** WORD is 20 bits. 0 Off 1 Sample data every WORD *(1 + DELAY) bits 2 Sample data every WORD *(1 + DELAY) + 1 bits 3 Sample data every WORD *(1 + DELAY) + 2 bits 4 Sample data every clk and assert XOR and MASK increment

49.5.51 PCIE_PHY_RX_SCOPE_PHASE

Address: 0h base + 1019h offset = 1019h

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved	BASE					SCOPE_DELAY	
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	SCOPE_ SEL	UPDATE	SAMPLE_PHASE					
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_RX_SCOPE_PHASE field descriptions

Field	Description
15 -	This field is reserved. Reserved
14–10 BASE	which bit to sample when MODE = 1 or 4
9–8 SCOPE_DELAY	How many clocks to delay the analog scope_data.
7 SCOPE_SEL	Select sampling mode. 0 Before AFE sampling 1 After AFE sampling
6 UPDATE	Update Sampling phase. Write a 1.
SAMPLE_ PHASE	Sampling Phase

49.5.52 PCIE_PHY_RX_DPLL_FREQ

Address: 0h base + 101Ah offset = 101Ah

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved			VAL				
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	VAL							DTHR
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_RX_DPLL_FREQ field descriptions

Field	Description
15–13 -	This field is reserved. Reserved
12–1 VAL	Freq is 1.526*VAL ppm from the reference When mpll_slow is set, the ppm is half the eqn above
0 DTHR	Bits below the useful resolution

49.5.53 PCIE_PHY_RX_CDR_CTL

Address: 0h base + 101Bh offset = 101Bh

Bit	15	14	13	12	11	10	9	8
Read Write	DTB_SEL				ALWAYS_ REALIGN	FAST_ START	FRUG_VALUE	
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	PHUG_VALUE		OVRD_ DPLL_GAIN	PHDET_ POL	PHDET_EDGE		PHDET_EN	
Reset	0	0	0	0	1	1	1	1

PCIE_PHY_RX_CDR_CTL field descriptions

Field	Description
15–12 DTB_SEL	Select to drive various signals onto the DTB. 0 disabled 1 pll_ana_rst,pll_count from rx_pwr_ctl 2 com_good_high/low from aligner 3 com_bad_high/low from aligner 4 shift_in_prog,ana_odd_data from aligner 5 Low bits of XAUI align FSM state
11 ALWAYS_ REALIGN	realign on any misaligned comma
10 FAST_START	decrease startup steps by 50%
9–8 FRUG_VALUE	override value for FRUG
7–6 PHUG_VALUE	override value for PHUG
5 OVRD_DPLL_ GAIN	Override PHUG and FRUG values
4 PHDET_POL	Reverse polarity of phase error

Table continues on the next page...

PCIE_PHY_RX_CDR_CTL field descriptions (continued)

Field	Description
3–2 PHDET_EDGE	Edges to use for phase detection. 11 Use both edges 10 Use rising edges only 01 Use falling edges only 00 Ignore all edges
PHDET_EN	Enables phase detecter. top bit is odd slicers, bottom is even

49.5.54 PCIE_PHY_RX_CDR_CDR_FSM_DEBUG

Address: 0h base + 101Ch offset = 101Ch

Bit	15	14	13	12	11	10	9	8
Read	adap_rx_eq			rx_eq_ctr			rx_ana_eq	
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	rx_ana_eq	adap_rx_ valid	cdr_en_ adap	cdr_en_eq	aligned	cdr_rx_valid	cdr_timeout	cdr_en
Write								
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_RX_CDR_CDR_FSM_DEBUG field descriptions

Field	Description
15–13 adap_rx_eq	Equalization setting from adaptation FSM.
12–10 rx_eq_ctr	Initial centre point from equalization FSM.
9–7 rx_ana_eq	Equalization setting to Analog.
6 adap_rx_valid	Adaptation has completed and locked
5 cdr_en_adap	Adapatation loop is enabling the CDR.
4 cdr_en_eq	Equalization loop is enabling the CDR.
3 aligned	Datapath is bit-aligned.
2 cdr_rx_valid	CDR has locked to incoming data stream.

Table continues on the next page...

PCIE_PHY_RX_CDR_CDR_FSM_DEBUG field descriptions (continued)

Field	Description
1 cdr_timeout	CDR has not locked to datastream and has timed-out.
0 cdr_en	CDR has been enabled.

49.5.55 PCIE_PHY_RX_CDR_LOCK_VEC_OVRD

Address: 0h base + 101Dh offset = 101Dh

Bit	15	14	13	12	11	10	9	8
Read Write	adap_ctr_level					adap_ polarity	lock_vector_ ovrd	lock_vector_ en
Reset	1	0	0	0	0	0	0	0

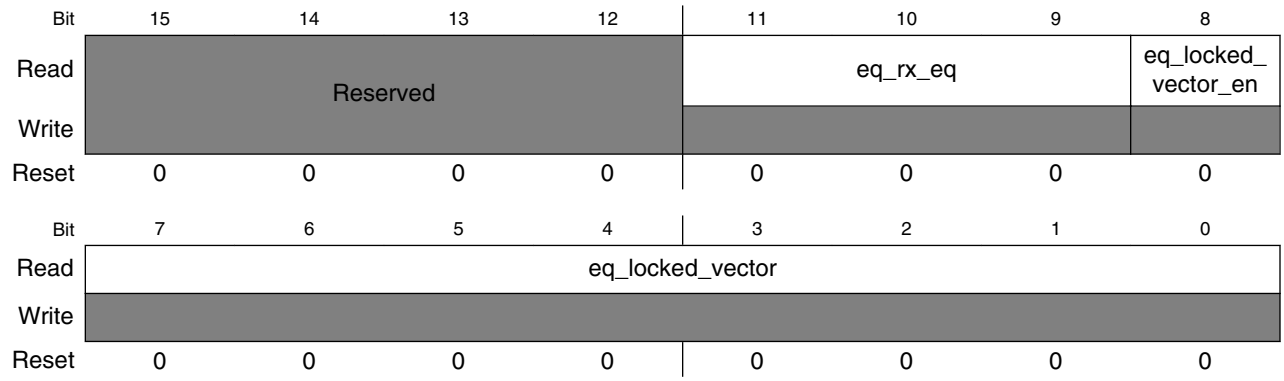
Bit	7	6	5	4	3	2	1	0
Read Write	lock_vector							
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_RX_CDR_LOCK_VEC_OVRD field descriptions

Field	Description
15–11 adap_ctr_level	Amount of earlies that increment the adaptation counter (times 16).
10 adap_polarity	If asserted invert default adaptation adjustment for equalization. IF early decrease equalization. Normal mode is to decrease.
9 lock_vector_ovrd	Override enable for the rx_eq outputs.
8 lock_vector_en	Override value for the locked_vector output completion.
lock_vector	Override value for the locked_vector.

49.5.56 PCIE_PHY_RX_CDR_LOCK_VEC

Address: 0h base + 101Eh offset = 101Eh

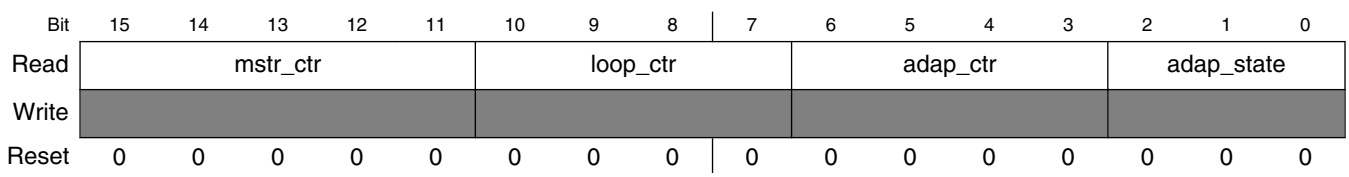


PCIE_PHY_RX_CDR_LOCK_VEC field descriptions

Field	Description
15–12 -	This field is reserved. Reserved
11–9 eq_rx_eq	Equalization setting from the Equalization Loop.
8 eq_locked_ vector_en	Equalization locked vector has been filled.
eq_locked_vector	Results of equalization loop.

49.5.57 PCIE_PHY_RX_CDR_ADAP_FSM

Address: 0h base + 101Fh offset = 101Fh



PCIE_PHY_RX_CDR_ADAP_FSM field descriptions

Field	Description
15–11 mstr_ctr	Master count register.
10–7 loop_ctr	Loop count register.
6–3 adap_ctr	Adaptation count register.

Table continues on the next page...

PCIE_PHY_RX_CDR_ADAP_FSM field descriptions (continued)

Field	Description
adap_state	Adaptation State. 000 ADAP_RESET 001 ADAP_LOCK 010 ADAP_SUFF 011 ADAP_LOOP 100 ADAP_MSTR 101 ADAP_DONE

49.5.58 PCIE_PHY_RX_ATB0

Address: 0h base + 1020h offset = 1020h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	Reserved								EN_ATB	EN_ATB	EN_MARG	EN_ATB_RM_F	EN_ATB_RM_S	EN_ATB_RP_F	EN_ATB_RP_S	EN_ATB_VOFF
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_PHY_RX_ATB0 field descriptions

Field	Description
15–8 -	This field is reserved. Reserved
7 EN_ATB	Enables ATB sensing and forcing on internal Rx nodes.
6 EN_ATB	Enables margining mode in receiver; requires atb_f_m to be high-Z!.
5 EN_MARG	Enables atb_force_p on negative-side termination resistor.
4 EN_ATB_RM_F	Enables atb_sense_m on negative-side termination resistor.
3 EN_ATB_RM_S	Enables atb_force_p on positive-side termination resistor.
2 EN_ATB_RP_F	Enables atb_sense_p on positive-side termination resistor.
1 EN_ATB_RP_S	Puts rxafe outputs vo_p on atb_s_p and vo_m on atb_s_m.
0 EN_ATB_VOFF	Puts rxafe voff_p on atb_s_p and voff_m on atb_s_m.

49.5.59 PCIE_PHY_RX_ATB1

Address: 0h base + 1021h offset = 1021h

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	VLOS_MAX	VLOS_MIN	EN_ATB_ VLOS	EN_ATB_ VRF	MEAS_GD	MEAS_VP	EN_VLOS_ USB3	NC0
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_RX_ATB1 field descriptions

Field	Description
15–8 -	This field is reserved. Reserved
7 VLOS_MAX	Sets LOS reference voltage. (VLOS_MAX, VLOS_MIN): (1,1): None (1,0): Maximum (0,1): Minimum (0,0): Nominal
6 VLOS_MIN	Sets LOS reference voltage. (VLOS_MAX, VLOS_MIN): (1,1): None (1,0): Maximum (0,1): Minimum (0,0): Nominal
5 EN_ATB_VLOS	Enables sensing of LOS reference voltage on atb_sense_p.
4 EN_ATB_VRF	Enables sensing of vref_rx on atb_sense_p.
3 MEAS_GD	Enables sensing of local gd in Rx; ties gd to atb_sense_m.
2 MEAS_VP	Enables sensing of local vp in Rx; ties vp to atb_sense_p.
1 EN_VLOS_USB3	Enables LOS levels to be those for USB3; otherwise, PCI Express levels.
0 NC0	Enables/disables Rx termination resistor.

49.5.60 PCIE_PHY_RX_ENPWR0

Address: 0h base + 1022h offset = 1022h

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	CTL_ RXPWRON	LCL_ RXPWRON	CTL_EN_ LOS	LCL_EN_ LOS	CTL_RXCK	LCL_RXCK	CTL_ACJT	LCL_ACJT
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_RX_ENPWR0 field descriptions

Field	Description
15–8 -	This field is reserved. Reserved
7 CTL_RXPWRON	Enables override of Rx block power.
6 LCL_RXPWRON	Enables/disables Rx slicers.
5 CTL_EN_LOS	Enables override of LOS block state.
4 LCL_EN_LOS	Enables/disables LOS block.
3 CTL_RXCK	Enables override of Rx clock circuit state.
2 LCL_RXCK	Enables/disables en_rx_clock (Rx clock enable).
1 CTL_ACJT	Enables override of ACJTAG block state.
0 LCL_ACJT	Enables/disables ACJTAG block.

49.5.61 PCIE_PHY_RX_PMIX_PHASE

Address: 0h base + 1023h offset = 1023h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	Reserved								PHASE							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_PHY_RX_PMIX_PHASE field descriptions

Field	Description
15–8 -	This field is reserved. Reserved
PHASE	Write to bits 8-1 of the Phase Select register in the phase mixer.

49.5.62 PCIE_PHY_RX_ENPWR1

Address: 0h base + 1024h offset = 1024h

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	LCL_ RXTERM	CTL_ RXTERM	LCL_BST			CTL_BST	LCL_ PHASE_ REG_RST	CTL_ PHASE_ REG_RST
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_RX_ENPWR1 field descriptions

Field	Description
15–8 -	This field is reserved. Reserved
7 LCL_RXTERM	Enables/disables Rx termination.
6 CTL_RXTERM	Enables override of rx_term_en.
5–3 LCL_BST	Rx boost (equalization) value
2 CTL_BST	Enables override of Rx boost (equalization) value.
1 LCL_PHASE_ REG_RST	Reset Phase register.
0 CTL_PHASE_ REG_RST	Enables override of Phase register reset.

49.5.63 PCIE_PHY_RX_ENPWR2

Address: 0h base + 1025h offset = 1025h

Bit	15	14	13	12	11	10	9	8
Read	Reserved							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	EN_RXPMIX_TST	EN_RXPMIX_VPMIX	EN_RXPMIX_VRX	EN_RXPMIX_VOSC	EN_RXPMIX_FRC_VPMIX	RX_SCOPE_ATB_2	RX_SCOPE_ATB_1	RX_SCOPE_ATB_0
Write								
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_RX_ENPWR2 field descriptions

Field	Description
15–8 -	This field is reserved. Reserved
7 EN_RXPMIX_TST	Enables XOR gate to test linearity of Rx phase mixer using atb_s_p and atb_s_m.
6 EN_RXPMIX_VPMIX	Puts vreg_pmix on atb_s_p.
5 EN_RXPMIX_VRX	Puts vreg_rx on atb_s_p.
4 EN_RXPMIX_VOSC	Puts vreg_vosc on atb_s_p.
3 EN_RXPMIX_FRC_VPMIX	Instructs rx_pmix_vreg_pmix to use atb_s_m as a reference instead of vbg.
2 RX_SCOPE_ATB_2	Puts XOR of Rx scope PMIX input and output on atb_s_p.
1 RX_SCOPE_ATB_1	Puts Rx scope regulated VP on atb_s_p.
0 RX_SCOPE_ATB_0	Instructs Rx scope regulated VP to use atb_f_p as reference instead of VP.

49.5.64 PCIE_PHY_RX_SCOPE

Address: 0h base + 1026h offset = 1026h

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	NC_SCOPE_2			RX_ SCOPE_ SLEW	RX_ SCOPE_ FDIV20	NC_SCOPE_3		
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_RX_SCOPE field descriptions

Field	Description
15–8 -	This field is reserved. Reserved
7–5 NC_SCOPE_2	NC
4 RX_SCOPE_ SLEW	Sets high for low Rx clock frqeuencies (625 MHz) for Rx scope to work correctly.
3 RX_SCOPE_ FDIV20	Divides scope output clock by 20 instead of 10.
NC_SCOPE_3	NC

49.5.65 PCIE_PHY_TX_TXDRV_CNTRL

Address: 0h base + 102Bh offset = 102Bh

Bit	15	14	13	12	11	10	9	8
Read	Reserved							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	PULL_DN_REG	PULL_UP_REG	OVRD_PULL_UP	VCM_HOLD_REG	OVRD_VCM_HOLD	Reserved	Reserved	Reserved
Write								
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_TX_TXDRV_CNTRL field descriptions

Field	Description
15–8 -	This field is reserved. Reserved
7 PULL_DN_REG	Register bit that causes the calibrated Tx bits to pull down in common mode fashion. If pull_dn_reg and tx_pull_up are both high, then pull_dn_reg wins (takes precedence").
6 PULL_UP_REG	Register override for tx_pull_up; selected when ovrd_pull_up is high; causes calibrated TX bits to pull up in common mode fashion, unless pull_dn_reg is high.
5 OVRD_PULL_UP	Selects loval value of pull_up_reg instead of tx_pull_up.
4 VCM_HOLD_REG	Register override for tx_vcm_hold; selected when ovrd_vcm_hold is high; controls the TX common mode hold circuitry.
3 OVRD_VCM_HOLD	Selects local value of vcm_hold_reg instead of tx_vcm_hold to control state of TX common mode hold circuitry.
2 NOCONN_8	This field is reserved. Reserved
1 NOCONN_7	This field is reserved. Reserved
0 NOCONN_6	This field is reserved. Reserved

49.5.66 PCIE_PHY_TX_POWER_CTL

Address: 0h base + 102Ch offset = 102Ch

Bit	15	14	13	12	11	10	9	8
Read	Reserved							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	OVRD_EN	SERIAL_ EN_REG	CLK_EN_ REG	DATA_EN_ REG	REFGEN_ EN_REG	TX_DIV_ CLK_EN	REFGEN_ PDN_REG	Reserved
Write								
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_TX_POWER_CTL field descriptions

Field	Description
15–8 -	This field is reserved. Reserved
7 OVRD_EN	Enables local overrides for all signals in this register.
6 SERIAL_EN_ REG	Value for tx_serial_en when OVRD_EN is 1.
5 CLK_EN_REG	Value for tx_clk_en when OVRD_EN is 1.
4 DATA_EN_REG	Value for tx_data_en when OVRD_EN is 1.
3 REFGEN_EN_ REG	Register override value for tx_refgen_en; turns on the pmos_bias refgen block and the rxdetect comparators.
2 TX_DIV_CLK_ EN	Enables the div clock that is output from the Tx to the undersampler, more appropriately called tx_sampler_clk_en; this clock is output after the optional divide-by-2/ 4; tx_clk_en must be high to output a clock.
1 REFGEN_PDN_ REG	Value for refgen_pwdn when OVRD_EN is 1.
0 NOCONN_5	This field is reserved. Reserved

49.5.67 PCIE_PHY_TX_ALT_BLOCK

Address: 0h base + 102Dh offset = 102Dh

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	EN_ALT_BUS	DRV_SOURCE_REG		JTAG_DATA_REG	ALT_OSC_VP	ALT_OSC_VPH	ALT_OSC_VPHREG	OVRD_ALT_BUS
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_TX_ALT_BLOCK field descriptions

Field	Description
15–8 -	This field is reserved. Reserved
7 EN_ALT_BUS	Enables the Tx for alt bus mode, powers up the pmos_bias block, and so on; required if manually running the alt bus features.
6–5 DRV_SOURCE_REG	Value for tx_data_source when OVRD_ALT_BUS is 1 11 JTAG data common mode for test 10 LFPS oscillator differential 01 JTAG data differential 00 Serializer data or alt oscillator vp/vph/vphreg/vptx if selected
4 JTAG_DATA_REG	Value for jtag_data when OVRD_ALT_BUS is 1.
3 ALT_OSC_VP	Enables and connects the vp oscillator to the transmit pins; must set drv_source_reg bus correctly.
2 ALT_OSC_VPH	Enables and connects the vph oscillator to the transmit pins; must set drv_source_reg bus correctly.
1 ALT_OSC_VPHREG	Enables and connects the vphreg oscillator to the transmit pins; must set drv_source_reg bus correctly.
0 OVRD_ALT_BUS	Enables local overrides for alt-bus control signals.

49.5.68 PCIE_PHY_TX_ALT_AND_LOOPBACK

Address: 0h base + 102Eh offset = 102Eh

Bit	15	14	13	12	11	10	9	8
Read	Reserved							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	OVRD_TX_LB	TX_LB_EN_REG	ALT_VPTX_OSC	Reserved	Reserved	Reserved	Reserved	Reserved
Write								
Reset	0	0	0	0	0	0	0	0

PCIE_PHY_TX_ALT_AND_LOOPBACK field descriptions

Field	Description
15–8 -	This field is reserved. Reserved
7 OVRD_TX_LB	Enables the override of the tx_lb_en pin.
6 TX_LB_EN_REG	Value of the tx_lb_en pin when OVRD_TX_LB is enabled.
5 ALT_VPTX_OSC	Enables and connects the vptx oscillator to the transmit pins; must set drv_source_reg bus correctly.
4 NOCONN_04	This field is reserved. Reserved
3 NOCONN_03	This field is reserved. Reserved
2 NOCONN_02	This field is reserved. Reserved
1 NOCONN_01	This field is reserved. Reserved
0 NOCONN_00	This field is reserved. Reserved

49.5.69 PCIE_PHY_TX_TX_ATB_REG

Address: 0h base + 102Fh offset = 102Fh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	Reserved								ATB_PBIAS	ATB_VCM_REP	ATB_RXDETREF	ATB_TXFP	ATB_TXFM	ATB_TXSP	ATB_TXSM	ATB_VCM
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCIE_PHY_TX_TX_ATB_REG field descriptions

Field	Description
15–8 -	This field is reserved. Reserved
7 ATB_PBIAS	Connects real pmos_bias voltage for Tx PMOS driver pull-up path to atb_s_p and local ground at the pmos_bias block to atb_s_m.
6 ATB_VCM_REP	Connects common mode replica voltage in pmos_bias block to atb_s_p and local ground to atb_s_m.
5 ATB_RXDETREF	Connects Rx detect block reference voltage to atb_s_p and local ground to atb_s_m.
4 ATB_TXFP	Connects tx_p to atb_f_p.
3 ATB_TXFM	Connects tx_m to atb_f_m.
2 ATB_TXSP	Connects tx_p to atb_s_p.
1 ATB_TXSM	Connects tx_m to atb_s_m.
0 ATB_VCM	Connects tx_p/tx_m common mode voltage onto atb_s_p and local ground onto atb_s_m.

Chapter 50
Power Management Unit (PMU)

50.1 Overview

The power management unit (PMU) is designed to simplify the external power interface. The power system can be split into the input power sources and their characteristics, the integrated power transforming and controlling elements, and the final load interconnection and requirements.

A typical power system utilizing the PMU is depicted below.

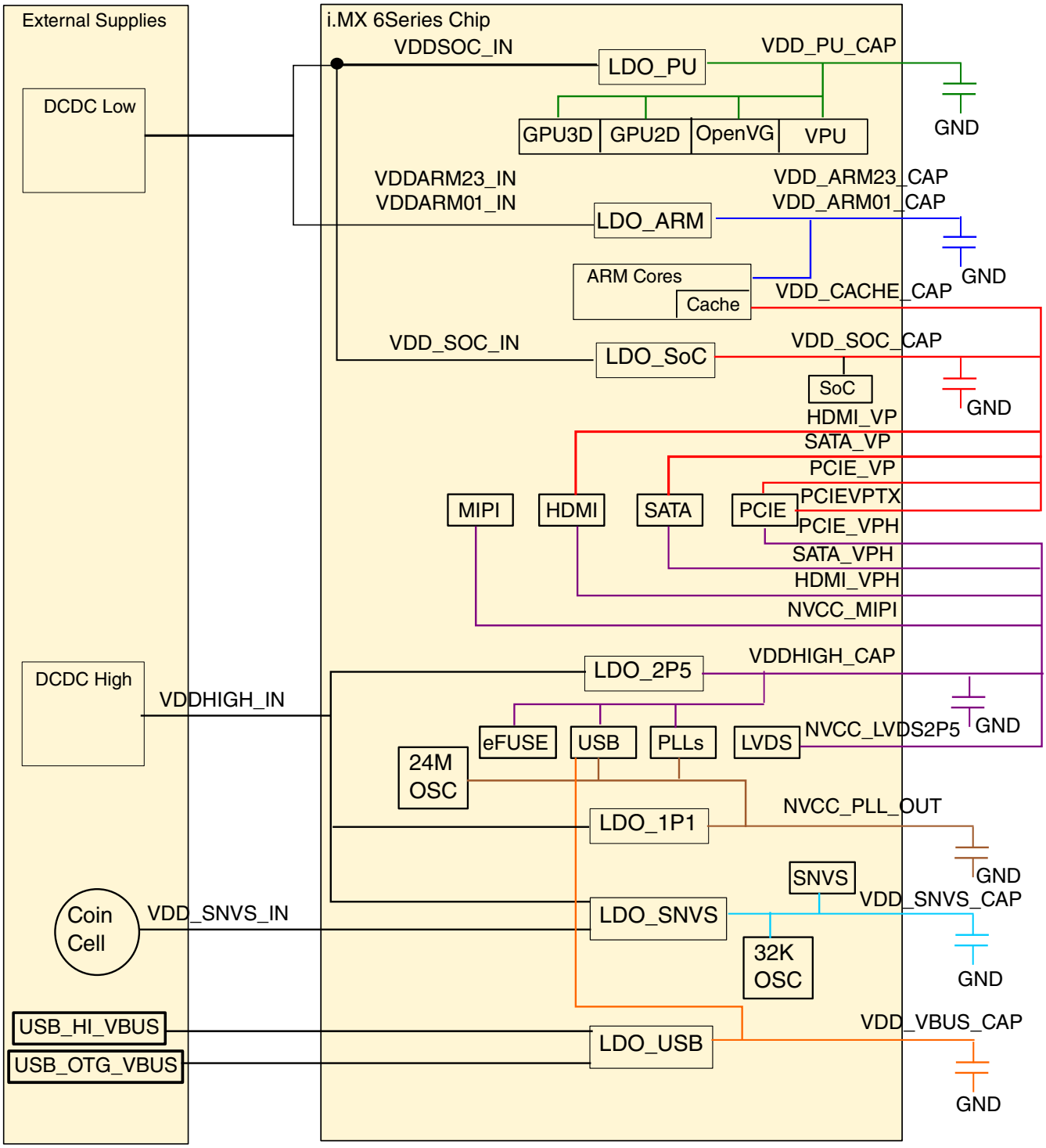


Figure 50-1. Power system overview

Utilizing seven LDO regulators, the number of external supplies is greatly reduced. Not counting the backup coin and USB inputs, the number of external supplies is reduced to two. Missing from this external supply total is the number of necessary external supplies to power the desired memory interface; that number varies depending on the type of external memory selected. Other supplies may also be necessary to supply the voltage to the different I/O power segments if their I/O voltages have to be different from what is provided above.

50.2 Digital LDO Regulators

The PMU has three digital LDO regulators. They are referred to as "digital" because of the logic loads they drive, not because of their construction. These regulators have three basic modes that are unique to the digital regulators.

- Internal Bypass—The regulation FET is switched fully on passing the external input voltage to the load unaltered. The analog part of the regulator is powered down in this state, removing any loss other than the IR drop through the power grid and the FET. (TARG = 0x1F)
- External Bypass—The input and output of the regulator are shorted externally to the part. If operating in this configuration, enable the internal bypass early in the start-up sequence before attempting high-frequency/high-power operation.
- Power Gate—The regulation FET is switched off fully, limiting the current draw from the supply. The analog part of the regulator is powered down, limiting the power consumption. The output voltage falls to a level at which the residual leakage of the power FET balances with the leakage of the load. (TARG = 0x00)
- Analog regulation mode—The regulation FET is controlled such that the output voltage of the regulator equals the programmed target voltage. The target voltage is fully programmable in 25-mV steps.

These modes allow the regulators to implement voltage scaling and power gating and allow bypass. With the bypass feature, all of the accuracy and control requirements can be shifted to the external supply source if capable and desired.

These digital regulators also feature brownout detection which is helpful when supplies are starting to collapse. The voltage value where brownout is signaled is programmable as an offset from the programmed target voltage. The controls are located in the PMU_MISC2 register. The core is interrupted on a brownout.

The three digital regulators are known as LDO_ARM, LDO_PU, and LDO_SOC. As shown in the power system overview figure, the ARM regulator powers the ARM cores. The LDO_PU powers the GPU, VPU, and display portions of the chip. The SOC regulator powers the rest of the digital logic on the chip. All regulators support generous

programming ranges in 25-mV steps. It is possible to program voltages above the process limit for the chip, thus causing permanent damage. Likewise, it is possible to program the voltage so low that the chip cannot continue to operate or even retain state without clocks. Care should be taken with these settings.

Care must be taken when raising the output voltage of the regulator rapidly. This can cause large currents to flow into the output cap of the regulator up to the limits of the input supply. When the input supply capability is exceeded, this can cause an input supply dip that may affect other regulators on the same supply. Therefore, the rate of voltage change on the output of the regulator should be limited. When powering up the regulator, the integrated current limiter controls the ramp rate. This limiter is only effective when transitioning from the off state of the regulator (bypassed or power gated).

However, in a DVFS situation, the same high rate of change can occur if the target voltage is raised rapidly by software. To limit the rate of change, the hardware controlling the regulator effects a piecewise linear ramp by stepping the output voltage in 25-mV steps until the desired output voltage is reached. The slope of the ramp is controlled by the time spent at each 25-mV step and is controlled by the step time field in the PMU_MISC2 register. The same situation is not a problem when the output voltage is dropped as the load pulls down the output cap. As a result, any reduction in the programmed regulator target voltage is immediately effective with the actual supply voltage falling at a rate controlled by the load on the regulator.

50.3 Analog LDO Regulators

There are two analog regulators described here.

50.3.1 LDO 1P1

The LDO_1P1 module on the chip implements a programmable linear-regulator function from a higher analog supply voltage (2.8 V–3.3 V) to produce a nominal 1.1-V output voltage.

The output of the regulator can be programmed in 25-mV steps from 0.8 V to 1.4 V. The regulator has been designed to be stable with a minimum external low-ESR decoupling capacitor of 4.7 µF, though the actual capacitance required should be determined by the application. A programmable brownout detector is included in the regulator which can be used by the system to determine when the load capability of the regulator is being exceeded, so the necessary steps can be taken.

Current limiting can be enabled by setting the PMU_REG_1P1[ENABLE_ILIMIT] bit to allow for in-rush current requirements during startup if needed. Active pulldown can also be enabled by setting the PMU_REG_1P1[ENABLE_PULLDOWN] bit for systems requiring this feature.

50.3.2 LDO 2P5

The LDO_2P5 module on the chip implements a programmable linear-regulator function from a higher analog supply voltage (2.8V-3.3V) to produce a nominal 2.5V output voltage.

The output of the regulator can be programmed in 25mV steps from 2.0V to 2.75V. The regulator has been designed to be stable with a minimum external low-ESR decoupling capacitor of 4.7μF, though the actual capacitance required should be determined by the application. A programmable brown-out detector is included in the regulator which can be used by the system to determine when the load capability of the regulator is being exceeded to take the necessary steps.

Current-limiting can be enabled by setting the REG_PMU_2P5[ENABLE_ILIMIT] bit to allow for in-rush current requirements during start-up if needed. Active-pulldown can also be enabled by setting the REG_PMU_2P5[ENABLE_PULLDOWN] bit for systems requiring this feature.

50.3.3 Low Power Operation

The 2.5 V LDO includes an alternate, self-biased, low-precision, weak regulator which can be enabled for applications needing to keep the 2.5-V output voltage alive during low-power modes where the main regulator and its associated global bandgap reference module are disabled.

The output of this weak regulator is not programmable and is a function of its input power supply as well as load current. The low-power mode is enabled by setting high the PMU_REG_2P5[ENABLE_WEAK_LINREG] bit of the regulator. It is recommended that the following sequence be followed to enable this mode:

1. Throttle down the 2.5 V attached load to its low-power maintain state.
2. Disable the main 2.5 V regulator driver by clearing the PMU_REG_2P5[ENABLE_LINREG] bit.
3. Enable the weak 2.5 V regulator by setting the PMU_REG_2P5[ENABLE_WEAK_LINREG] bit.

To go back to full-power operation, reverse the steps outlined above. Note that the external decoupling cap is supporting the power supply between steps 2 and 3. Therefore step 3 should happen appropriately in time relative to the discharge of the supporting capacitor.

50.4 USB LDO Regulator

The USB_LDO module on the chip implements a programmable linear-regulator function from the USB VBUS voltages (typically 5 V) to produce a nominal 3.0-V output voltage.

The output of the regulator can be programmed in 25-mV steps, from 2.625V to 3.4 V . The regulator has been designed to be stable with a minimum external low-ESR decoupling capacitor of 4.7 μF, though the actual capacitance required should be determined by the application. A programmable brownout detector is included in the regulator which can be used by the system to determine when the load capability of the regulator is being exceeded, so the necessary steps can be taken. This regulator has a built-in power mux which allows the user to choose to run the regulator from either VBUS supply when both are present. If only one of the VBUS voltages is present, then the regulator automatically selects this supply. Current limit is also included to help the system meet in-rush current targets.

Upon attachment of VBUS, this regulator starts up in a low-power, self-preservation mode to prevent over-voltage conditions on the chip. It is expected that the user transition to full regulation by enabling the regulator and disabling the in-rush current limits via its control registers. Upon VBUS removal, it is further expected that the regulator controls are returned to their reset state.

50.5 SNVS Regulator

The SNVS regulator takes the SNVS_IN supply and generates the SNVS_CAP supply, which powers the real time clock and SNVS blocks.

If VDDHIGH_IN is present, then the SNVS_IN supply is internally shorted to the VDDHIGH_IN supply to allow coin cell recharging if necessary. The output voltage is roughly one third of SNVS_IN.

50.6 Power Modes

50.6.1 Reverse Well Biasing

The reverse well biasing module on the chip includes a self-clocked/self-regulating charge-pump circuit to generate a negative bias voltage for the floating PWELL, and a low-power regulator to generate a positive bias voltage for the NWELL of digital logic cells on the SOC power domain.

Static leakage reduction can be achieved through the use of these reverse well bias voltages. Typical power consumption of the module is 50 µA when driving a 10-nF purely capacitive load.

50.7 PMU Memory Map/Register Definition

The register definitions that affect the behavior of the digital LDO regulators follow.

NOTE

Some of the registers are collections of bits that affect multiple components on the chip. Those that are not pertinent to this chapter have comments in the related register bitfields.

If a full description is desired, please consult the full register programming reference in the related block.

PMU memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20C_8110	Regulator 1P1 Register (PMU_REG_1P1)	32	R/W	0000_1073h	50.7.1/4439
20C_8120	Regulator 3P0 Register (PMU_REG_3P0)	32	R/W	0000_0F74h	50.7.2/4442
20C_8130	Regulator 2P5 Register (PMU_REG_2P5)	32	R/W	0000_5071h	50.7.3/4444
20C_8140	Digital Regulator Core Register (PMU_REG_CORE)	32	R/W	0040_2010h	50.7.4/4446
20C_8150	Miscellaneous Register 0 (PMU_MISC0)	32	R/W	0400_0000h	50.7.5/4449
20C_8160	Miscellaneous Register 1 (PMU_MISC1)	32	R/W	0000_0000h	50.7.6/4452
20C_8164	Miscellaneous Register 1 (PMU_MISC1_SET)	32	R/W	0000_0000h	50.7.6/4452
20C_8168	Miscellaneous Register 1 (PMU_MISC1_CLR)	32	R/W	0000_0000h	50.7.6/4452
20C_816C	Miscellaneous Register 1 (PMU_MISC1_TOG)	32	R/W	0000_0000h	50.7.6/4452
20C_8170	Miscellaneous Control Register (PMU_MISC2)	32	R/W	0027_2727h	50.7.7/4455
20C_8174	Miscellaneous Control Register (PMU_MISC2_SET)	32	R/W	0027_2727h	50.7.7/4455

Table continues on the next page...

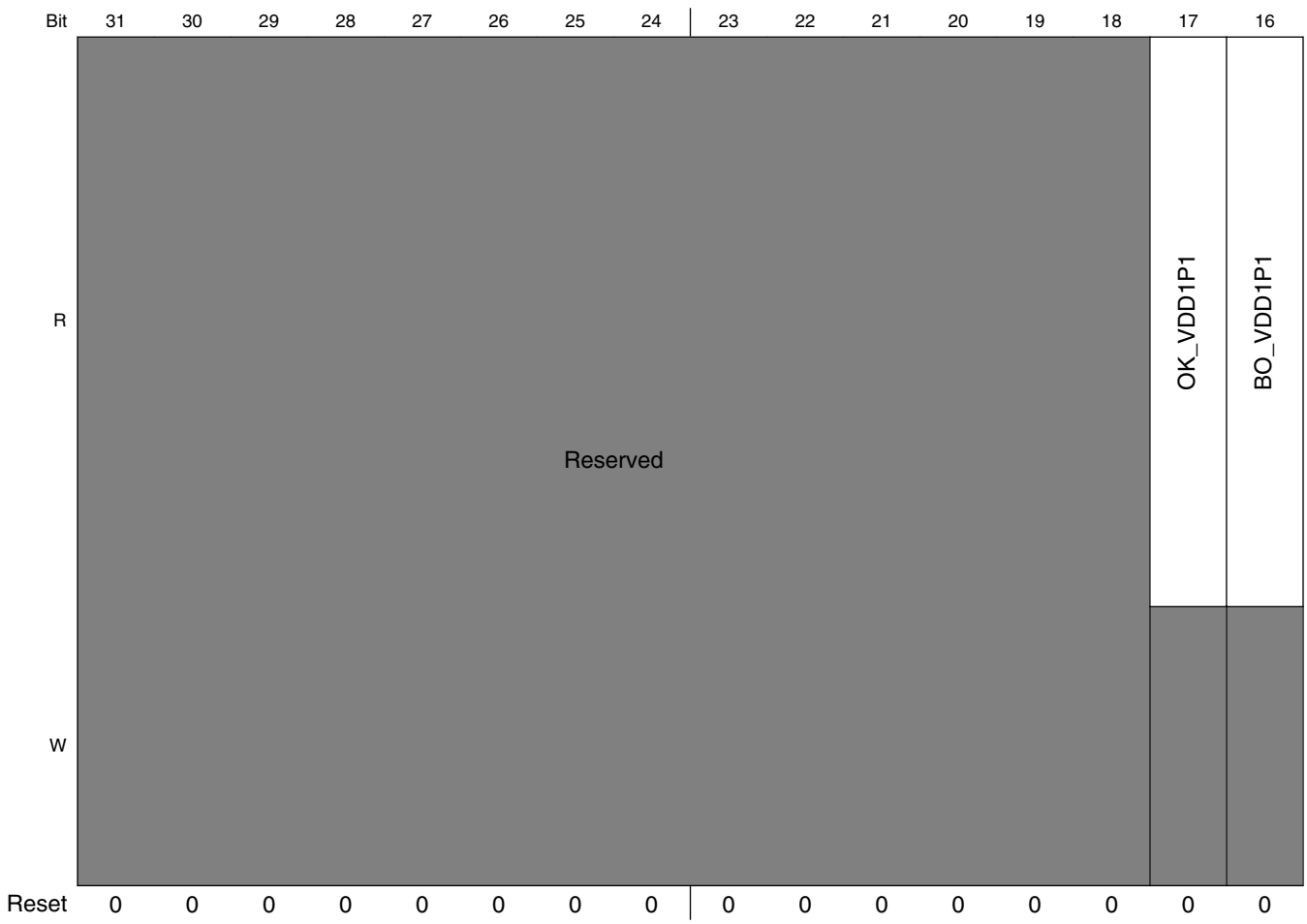
PMU memory map (continued)

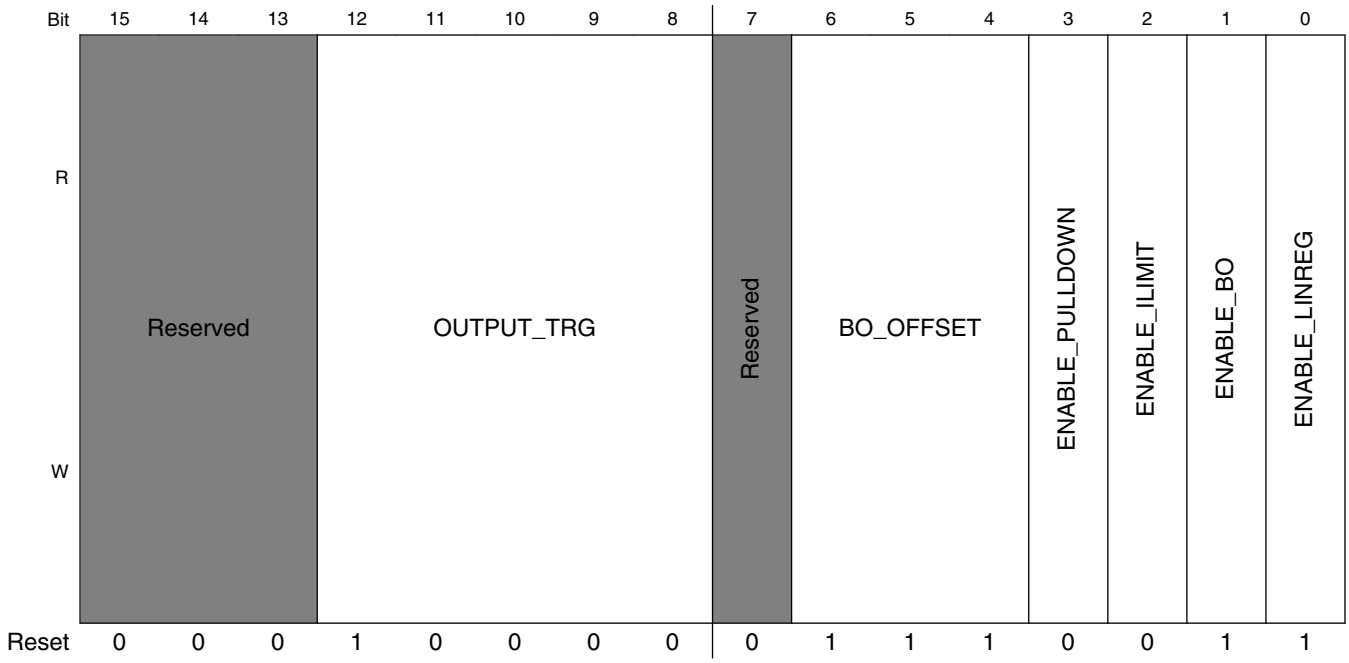
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20C_8178	Miscellaneous Control Register (PMU_MISC2_CLR)	32	R/W	0027_2727h	50.7.7/4455
20C_817C	Miscellaneous Control Register (PMU_MISC2_TOG)	32	R/W	0027_2727h	50.7.7/4455

50.7.1 Regulator 1P1 Register (PMU_REG_1P1)

This register defines the control and status bits for the 1.1V regulator. This regulator is designed to power the digital portions of the analog cells.

Address: 20C_8000h base + 110h offset = 20C_8110h





PMU_REG_1P1 field descriptions

Field	Description
31–18 -	This field is reserved.
17 OK_VDD1P1	Status bit that signals when the regulator output is ok. 1 = regulator output > brownout target
16 BO_VDD1P1	Status bit that signals when a brownout is detected on the regulator output.
15–13 -	This field is reserved.
12–8 OUTPUT_TRG	Control bits to adjust the regulator output voltage. Each LSB is worth 25mV. Programming examples are detailed below. Other output target voltages may be interpolated from these examples. Choices must be in this range: 0x1b >= output_trg >= 0x04 **NOTE:** There may be reduced chip functionality or reliability at the extremes of the programming range. 0x04 0.8V 0x10 1.1V 0x1b 1.375V
7 -	This field is reserved.
6–4 BO_OFFSET	Control bits to adjust the regulator brownout offset voltage in 25mV steps. The reset brown-offset is 175mV below the programmed target code. Brownout target = OUTPUT_TRG - BO_OFFSET. Some steps may be irrelevant because of input supply limitations or load operation.
3 ENABLE_ PULLDOWN	Control bit to enable the pull-down circuitry in the regulator
2 ENABLE_ILIMIT	Control bit to enable the current-limit circuitry in the regulator.

Table continues on the next page...

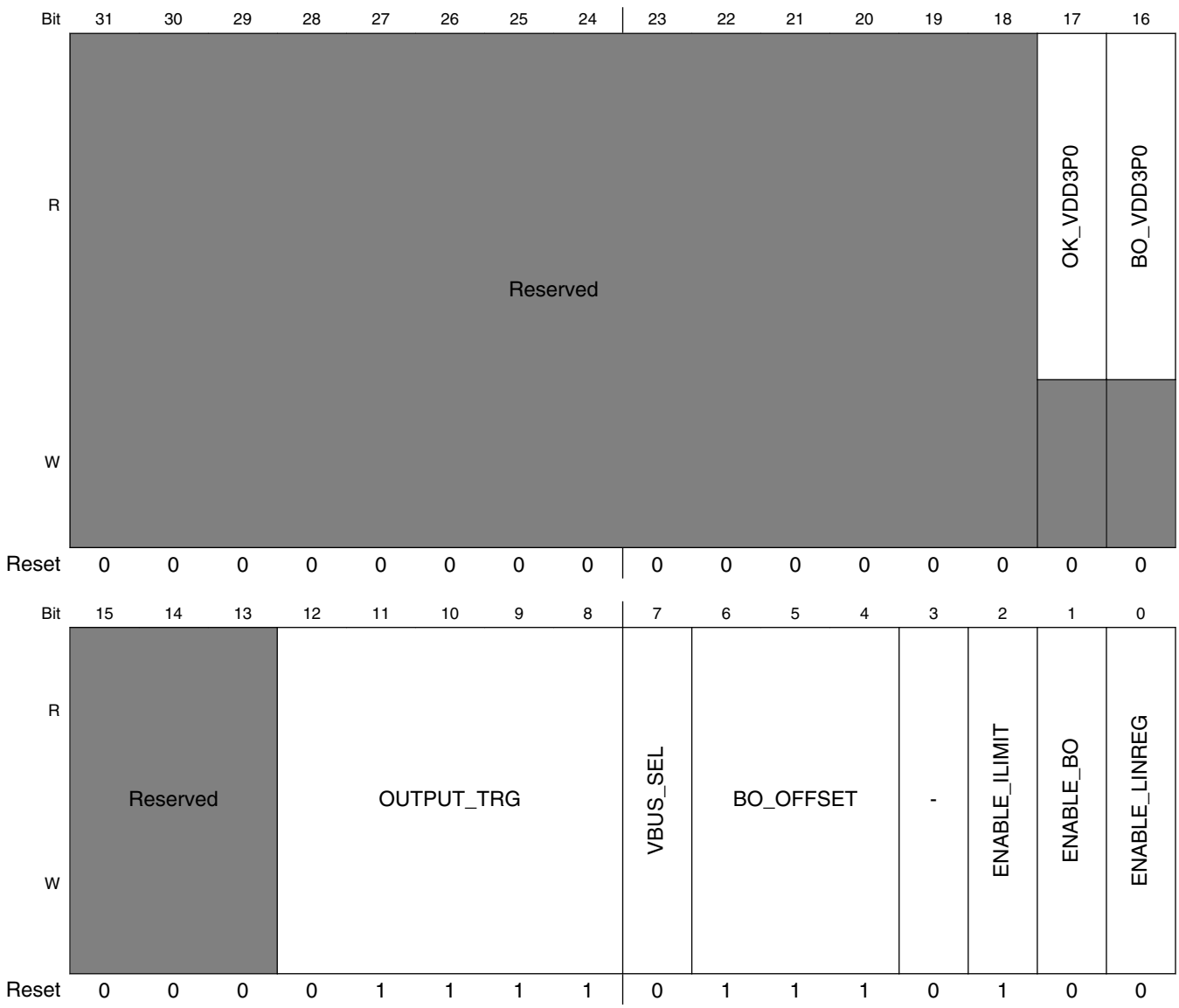
PMU_REG_1P1 field descriptions (continued)

Field	Description
1 ENABLE_BO	Control bit to enable the brownout circuitry in the regulator.
0 ENABLE_ LINREG	Control bit to enable the regulator output.

50.7.2 Regulator 3P0 Register (PMU_REG_3P0)

This register defines the control and status bits for the 3.0V regulator powered by the host USB VBUS pin.

Address: 20C_8000h base + 120h offset = 20C_8120h



PMU_REG_3P0 field descriptions

Field	Description
31–18 -	This field is reserved.

Table continues on the next page...

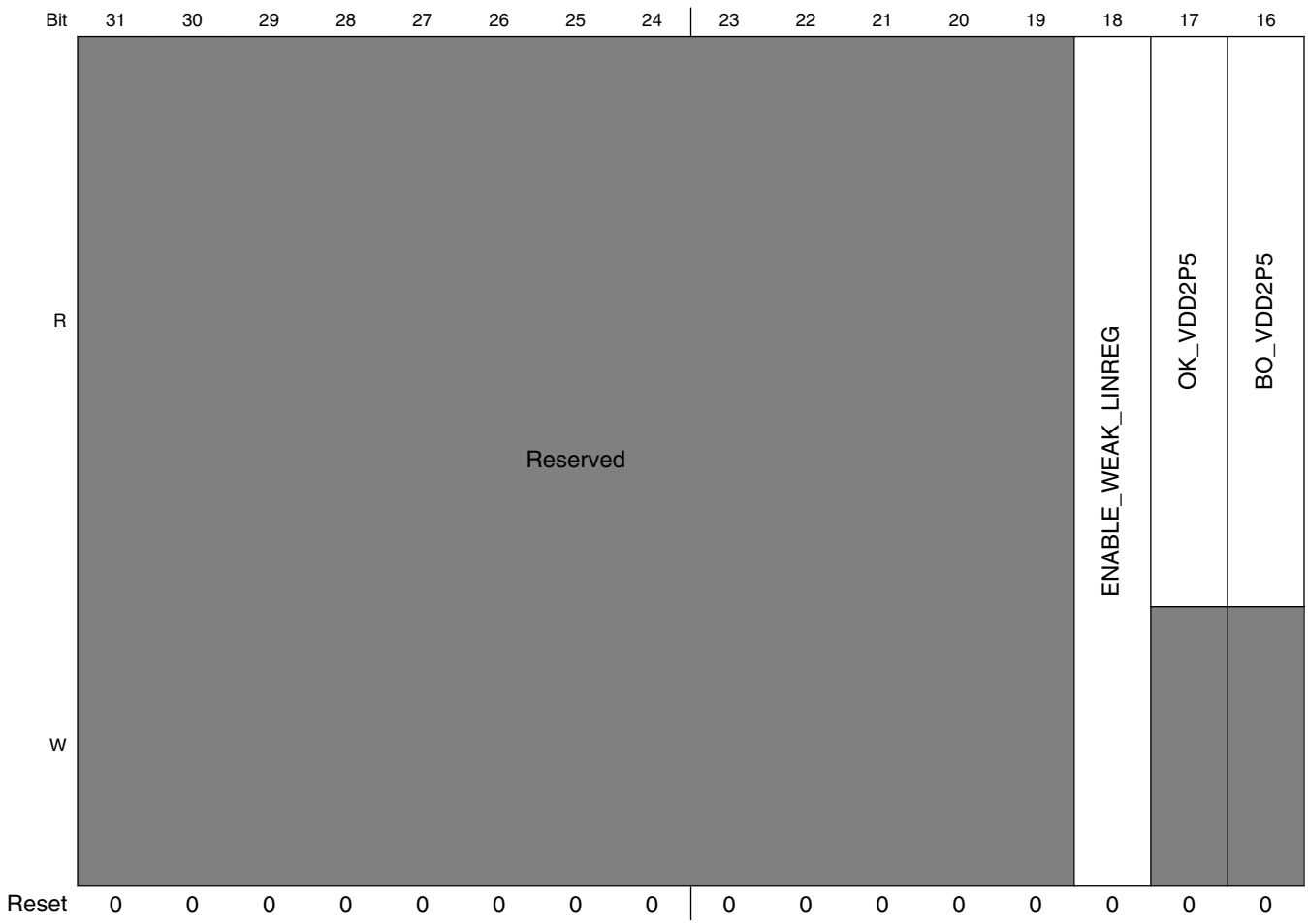
PMU_REG_3P0 field descriptions (continued)

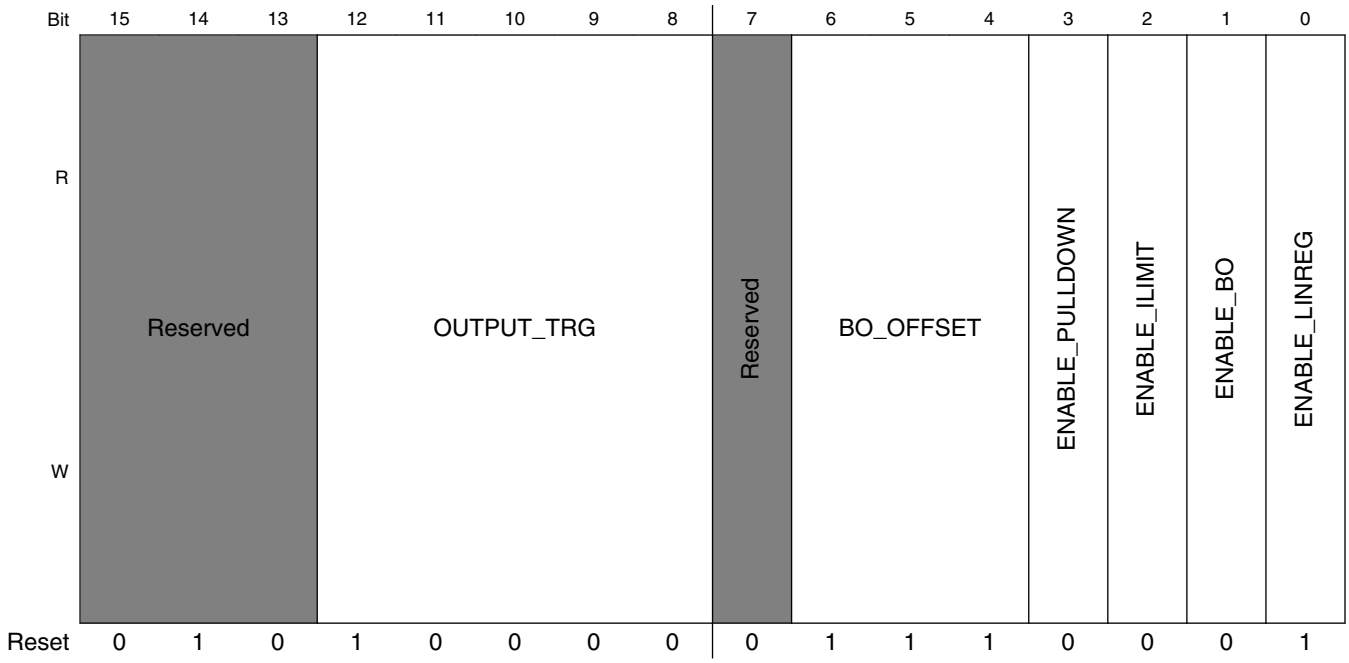
Field	Description
17 OK_VDD3P0	Status bit that signals when the regulator output is ok. 1 = regulator output > brownout target
16 BO_VDD3P0	Status bit that signals when a brownout is detected on the regulator output.
15–13 -	This field is reserved.
12–8 OUTPUT_TRG	Control bits to adjust the regulator output voltage. Each LSB is worth 25mV. Programming examples are detailed below. Other output target voltages may be interpolated from these examples. **NOTE:** There may be reduced chip functionality or reliability at the extremes of the programming range. 0x00 2.625V 0x0f 3.000V 0x1f 3.400V
7 VBUS_SEL	Select input voltage source for LDO_3P0 from either USB_H1_VBUS or USB_OTG_VBUS. If only one of the two VBUS voltages is present, it will automatically be selected. 0 **USB_H1_VBUS** — Utilize VBUS H1 for power 1 **USB_OTG_VBUS** — Utilize VBUS OTG for power
6–4 BO_OFFSET	Control bits to adjust the regulator brownout offset voltage in 25mV steps. The reset brown-offset is 175mV below the programmed target code. Brownout target = OUTPUT_TRG - BO_OFFSET. Some steps may not be relevant because of input supply limitations or load operation.
3 -	Reserved
2 ENABLE_ILIMIT	Control bit to enable the current-limit circuitry in the regulator.
1 ENABLE_BO	Control bit to enable the brownout circuitry in the regulator.
0 ENABLE_ LINREG	Control bit to enable the regulator output to be set by the programmed target voltage setting and internal bandgap reference.

50.7.3 Regulator 2P5 Register (PMU_REG_2P5)

This register defines the control and status bits for the 2.5V regulator.

Address: 20C_8000h base + 130h offset = 20C_8130h





PMU_REG_2P5 field descriptions

Field	Description
31–19 -	This field is reserved.
18 ENABLE_ WEAK_LINREG	Enables the weak 2p5 regulator. This low power regulator is used when the main 2p5 regulator is disabled to keep the 2.5V output roughly at 2.5V. Scales directly with the value of VDDHIGH_IN.
17 OK_VDD2P5	Status bit that signals when the regulator output is ok. 1 = regulator output > brownout target
16 BO_VDD2P5	Status bit that signals when a brownout is detected on the regulator output.
15–13 -	This field is reserved.
12–8 OUTPUT_TRG	Control bits to adjust the regulator output voltage. Each LSB is worth 25mV. Programming examples are detailed below. Other output target voltages may be interpolated from these examples. **NOTE:** There may be reduced chip functionality or reliability at the extremes of the programming range. 0x00 2.10V 0x10 2.50V 0x1f 2.875V
7 -	This field is reserved.
6–4 BO_OFFSET	Control bits to adjust the regulator brownout offset voltage in 25mV steps. The reset brown-offset is 175mV below the programmed target code. Brownout target = OUTPUT_TRG - BO_OFFSET. Some steps may be irrelevant because of input supply limitations or load operation.
3 ENABLE_ PULLDOWN	Control bit to enable the pull-down circuitry in the regulator

Table continues on the next page...

PMU_REG_2P5 field descriptions (continued)

Field	Description
2 ENABLE_ILIMIT	Control bit to enable the current-limit circuitry in the regulator.
1 ENABLE_BO	Control bit to enable the brownout circuitry in the regulator.
0 ENABLE_ LINREG	Control bit to enable the regulator output.

50.7.4 Digital Regulator Core Register (PMU_REG_CORE)

This register defines the function of the digital regulators

Address: 20C_8000h base + 140h offset = 20C_8140h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved		FET_ODRIVE	Reserved					REG2_TARG						Reserved	
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved		REG1_TARG						Reserved			REG0_TARG				
Reset	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0

PMU_REG_CORE field descriptions

Field	Description
31–30 -	This field is reserved.
29 FET_ODRIVE	If set, increases the gate drive on power gating FETs to reduce leakage in the off state. Care must be taken to apply this bit only when the input supply voltage to the power FET is less than 1.1V. **NOTE:** This bit should only be used in low-power modes where the external input supply voltage is nominally 0.9V.
28–23 -	This field is reserved.

Table continues on the next page...

PMU_REG_CORE field descriptions (continued)

Field	Description
22–18 REG2_TARG	This field defines the target voltage for the SOC power domain. Single-bit increments reflect 25mV core voltage steps. Some steps may not be relevant because of input supply limitations or load operation. **NOTE:** This register is capable of programming an over-voltage condition on the device. Consult the datasheet Operating Ranges table for the allowed voltages. 00000 Power gated off 00001 Target core voltage = 0.725V 00010 Target core voltage = 0.750V 00011 Target core voltage = 0.775V ... 10000 Target core voltage = 1.100V ... 11110 Target core voltage = 1.450V 11111 Power FET switched full on. No regulation.
17–14 -	This field is reserved.
13–9 REG1_TARG	This field defines the target voltage for the VPU/GPU power domain. Single-bit increments reflect 25mV core voltage steps. Some steps may not be relevant because of input supply limitations or load operation. **NOTE:** This register is capable of programming an over-voltage condition on the device. Consult the datasheet Operating Ranges table for the allowed voltages. 00000 Power gated off 00001 Target core voltage = 0.725V 00010 Target core voltage = 0.750V 00011 Target core voltage = 0.775V ... 10000 Target core voltage = 1.100V ... 11110 Target core voltage = 1.450V 11111 Power FET switched full on. No regulation.
8–5 -	This field is reserved.
REG0_TARG	This field defines the target voltage for the ARM core power domain. Single-bit increments reflect 25mV core voltage steps. Some steps may not be relevant because of input supply limitations or load operation. **NOTE:** This register is capable of programming an over-voltage condition on the device. Consult the datasheet Operating Ranges table for the allowed voltages. 00000 Power gated off 00001 Target core voltage = 0.725V 00010 Target core voltage = 0.750V 00011 Target core voltage = 0.775V ... 10000 Target core voltage = 1.100V ...

Table continues on the next page...

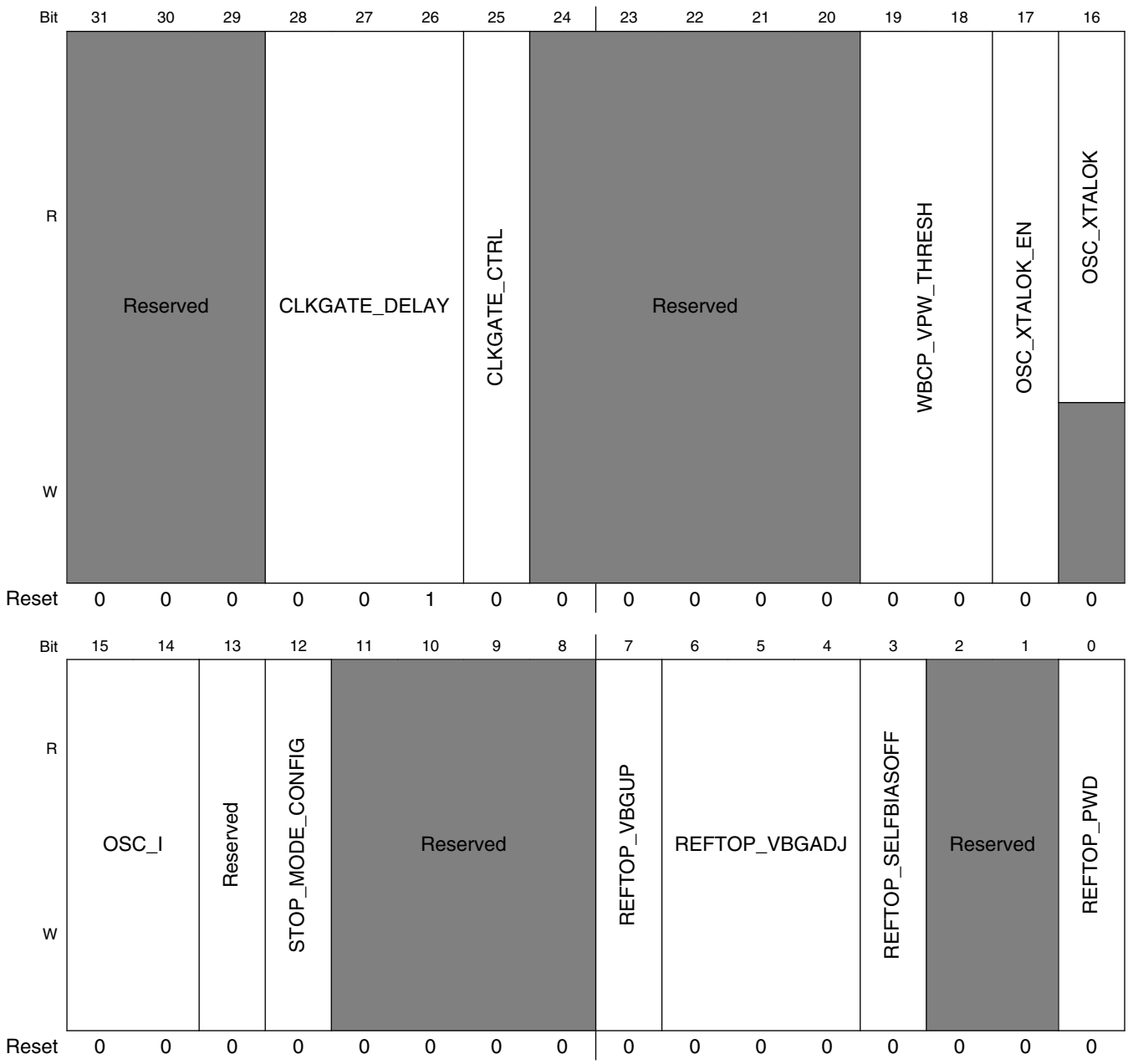
PMU_REG_CORE field descriptions (continued)

Field	Description
	11110 Target core voltage = 1.450V 11111 Power FET switched full on. No regulation.

50.7.5 Miscellaneous Register 0 (PMU_MISC0)

This register defines the control and status bits for miscellaneous analog blocks.

Address: 20C_8000h base + 150h offset = 20C_8150h



PMU_MISC0 field descriptions

Field	Description
31–29 -	This field is reserved.
28–26 CLKGATE_ DELAY	This field specifies the delay between powering up the XTAL 24MHz clock and releasing the clock to the digital logic inside the analog block. **NOTE:** Do not change the field during a low power event. This is not a field that the user would normally need to modify. **NOTE:** Not related to PMU. 000 0.5ms 001 1.0ms 010 2.0ms 011 3.0ms 100 4.0ms 101 5.0ms 110 6.0ms 111 7.0ms
25 CLKGATE_CTRL	This bit allows disabling the clock gate (always ungated) for the xtal 24MHz clock that clocks the digital logic in the analog block. **NOTE:** Do not change the field during a low power event. This is not a field that the user would normally need to modify. **NOTE:** Not related to PMU. 0 **ALLOW_AUTO_GATE** — Allow the logic to automatically gate the clock when the XTAL is powered down. 1 **NO_AUTO_GATE** — Prevent the logic from ever gating off the clock.
24–20 -	This field is reserved. Always set to zero.
19–18 WBCP_VPW_ THRESH	This signal alters the voltage that the pwell is charged pumped to. 00 **NOMINAL_BIAS** — Nominal output pwell bias voltage. 01 **PLUS_25MV** — Increase pwell output voltage by 25mV. 10 **MINUS_25MV** — Decrease pwell output pwell voltage by 25mV. 11 **MINUS_50MV** — Decrease pwell output pwell voltage by 50mV.
17 OSC_XTALOK_ EN	This bit enables the detector that signals when the 24MHz crystal oscillator is stable. **NOTE:** Not related to PMU, Clocking content
16 OSC_XTALOK	Status bit that signals that the output of the 24-MHz crystal oscillator is stable. Generated from a timer and active detection of the actual frequency. **NOTE:** Not related to PMU, clocking content.
15–14 OSC_I	This field determines the bias current in the 24MHz oscillator. The aim is to start up with the highest bias current, which can be decreased after startup if it is determined to be acceptable. **NOTE:** Not related to PMU.

Table continues on the next page...

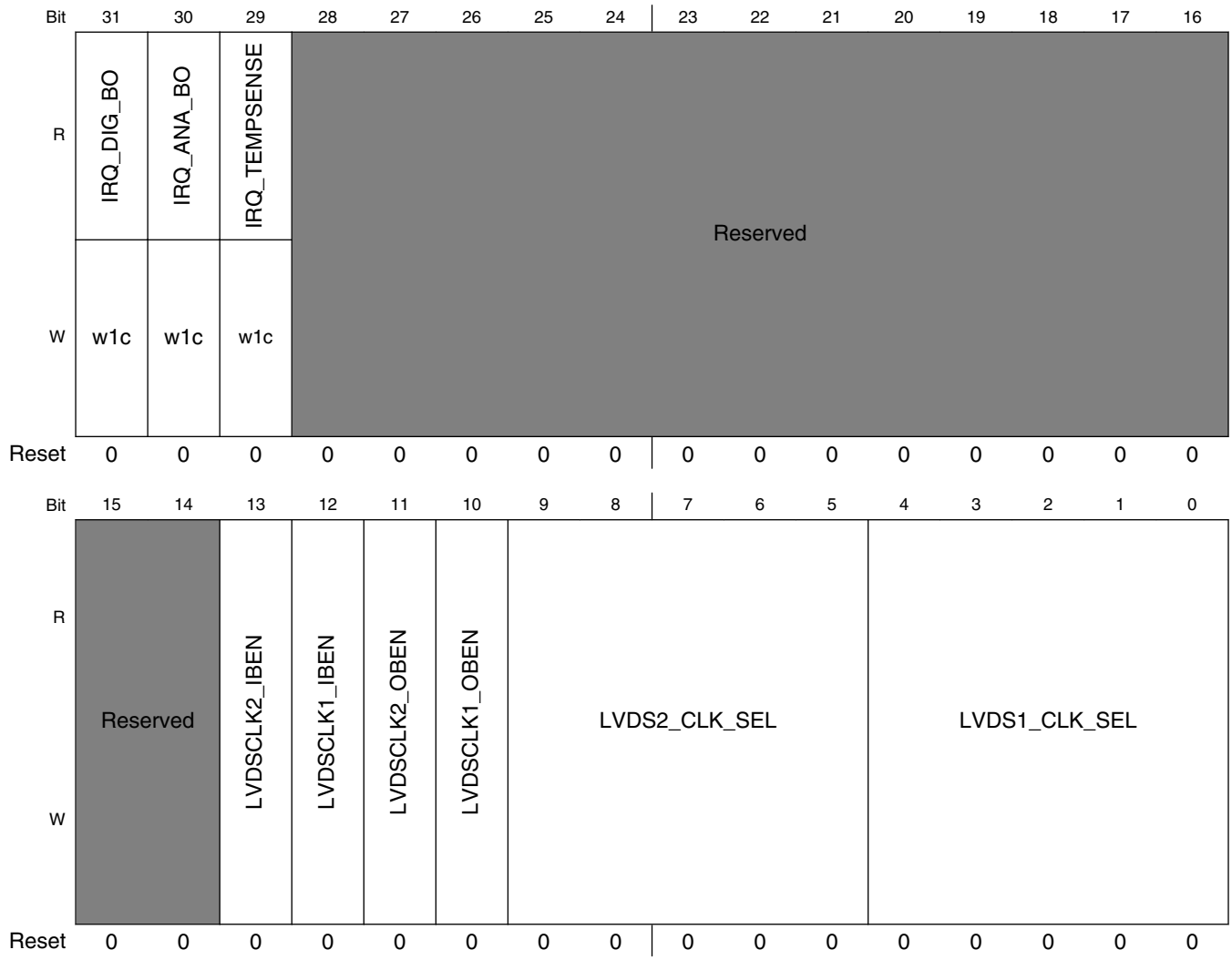
PMU_MISC0 field descriptions (continued)

Field	Description
	00 **NOMINAL** — Nominal 01 **MINUS_12_5_PERCENT** — Decrease current by 12.5% 10 **MINUS_25_PERCENT** — Decrease current by 25.0% 11 **MINUS_37_5_PERCENT** — Decrease current by 37.5%
13 Reserved	This field is reserved. Reserved
12 STOP_MODE_CONFIG	Configure the analog behavior in stop mode. 0x0 **DEEP** — Deep Stop Mode - 0x0 All analog except RTC powered down on Stop mode assertion 0x1 **LIGHT** — Light Stop Mode - 0x1 All the analog domain except the LDO_1P1, LDO_2P5, and PLL3 is powered down on STOP mode assertion. If required the CCM can be configured not to power down the oscillator (XTALOSC). PLL3 can be disabled with register settings if desired.
11–8 -	This field is reserved. Reserved
7 REFTOP_VBGUP	Status bit that signals the analog bandgap voltage is up and stable. 1 - Stable.
6–4 REFTOP_VBGADJ	000 Nominal VBG 001 VBG+0.78% 010 VBG+1.56% 011 VBG+2.34% 100 VBG-0.78% 101 VBG-1.56% 110 VBG-2.34% 111 VBG-3.12%
3 REFTOP_SELFBIASOFF	Control bit to disable the self-bias circuit in the analog bandgap. The self-bias circuit is used by the bandgap during startup. This bit should be set after the bandgap has stabilized and is necessary for best noise performance of analog blocks using the outputs of the bandgap. **NOTE:** Value should be returned to zero before removing vddhigh_in or asserting bit 0 of this register (REFTOP_PWD) to assure proper restart of the circuit. 0 Uses coarse bias currents for startup 1 Uses bandgap-based bias currents for best performance.
2–1 -	This field is reserved.
0 REFTOP_PWD	Control bit to power-down the analog bandgap reference circuitry. **NOTE:** A note of caution, the bandgap is necessary for correct operation of most of the LDO, pll, and other analog functions on the die.

50.7.6 Miscellaneous Register 1 (PMU_MISC1*n*)

This register defines the control and status bits for miscellaneous analog blocks. The LVDS1 and LVDS2 controls below control the behavior of the anaclk1/1b and anaclk2/2b LVDS IO's.

Address: 20C_8000h base + 160h offset + (4d × i), where i=0d to 3d



PMU_MISC1*n* field descriptions

Field	Description
31 IRQ_DIG_BO	This status bit is set to one when when any of the digital regulator brownout interrupts assert. Check the regulator status bits to discover which regulator interrupt asserted.
30 IRQ_ANA_BO	This status bit is set to one when when any of the analog regulator brownout interrupts assert. Check the regulator status bits to discover which regulator interrupt asserted.

Table continues on the next page...

PMU_MISC1*n* field descriptions (continued)

Field	Description
29 IRQ_ TEMPSENSE	This status bit is set to one when when the temperature sensor interrupt asserts. **NOTE:** Not related to PMU, Temperature Monitor content.
28–14 -	This field is reserved.
13 LVDSCLK2_ IBEN	This enables the LVDS input buffer for anaclk2/2b. Do not enable input and output buffers simultaneously. **NOTE:** Not related to PMU.
12 LVDSCLK1_ IBEN	This enables the LVDS input buffer for anaclk1/1b. Do not enable input and output buffers simultaneously. **NOTE:** Not related to PMU, Clocking content.
11 LVDSCLK2_ OBEN	This enables the LVDS output buffer for anaclk2/2b. Do not enable input and output buffers simultaneously. **NOTE:** Not related to PMU.
10 LVDSCLK1_ OBEN	This enables the LVDS output buffer for anaclk1/1b. Do not enable input and output buffers simultaneously. **NOTE:** Not related to PMU, clocking content.
9–5 LVDS2_CLK_ SEL	This field selects the clk to be routed to anaclk2/2b. **NOTE:** Not related to PMU. 00000 **ARM_PLL** — Arm PLL 00001 **SYS_PLL** — System PLL 00010 **PFD4** — pfd4 00011 **PFD5** — pfd5 00100 **PFD6** — pfd6 00101 **PFD7** — pfd7 00110 **AUDIO_PLL** — Audio PLL 00111 **VIDEO_PLL** — Video PLL 01000 **MLB_PLL** — MLB PLL 01001 **ETHERNET_REF** — ethernet ref clock 01010 **PCIE_REF** — PCIe ref clock 01011 **SATA_REF** — SATA ref clock 01100 **USB1_PLL** — USB1 PLL clock 01101 **USB2_PLL** — USB2 PLL clock 01110 **PFD0** — pfd0 01111 **PFD1** — pfd1 10000 **PFD2** — pfd2 10001 **PFD3** — pfd3 10010 **XTAL** — xtal 10011 **LVDS1** — LVDS1 (loopback) 10100 **LVDS2** — LVDS2 (not useful) 10101 to 11111 pfd7

Table continues on the next page...

PMU_MISC1*n* field descriptions (continued)

Field	Description
LVDS1_CLK_SEL	This field selects the clk to be routed to anaclk2/2b. **NOTE:** Not related to PMU. 00000 **ARM_PLL** — Arm PLL 00001 **SYS_PLL** — System PLL 00010 **PFD4** — pfd4 00011 **PFD5** — pfd5 00100 **PFD6** — pfd6 00101 **PFD7** — pfd7 00110 **AUDIO_PLL** — Audio PLL 00111 **VIDEO_PLL** — Video PLL 01000 **MLB_PLL** — MLB PLL 01001 **ETHERNET_REF** — ethernet ref clock 01010 **PCIE_REF** — PCIe ref clock 01011 **SATA_REF** — SATA ref clock 01100 **USB1_PLL** — USB1 PLL clock 01101 **USB2_PLL** — USB2 PLL clock 01110 **PFD0** — pfd0 01111 **PFD1** — pfd1 10000 **PFD2** — pfd2 10001 **PFD3** — pfd3 10010 **XTAL** — xtal 10011 **LVDS1** — LVDS1 (loopback) 10100 **LVDS2** — LVDS2 (not useful) 10101 to 11111 pfd7

50.7.7 Miscellaneous Control Register (PMU_MISC2*n*)

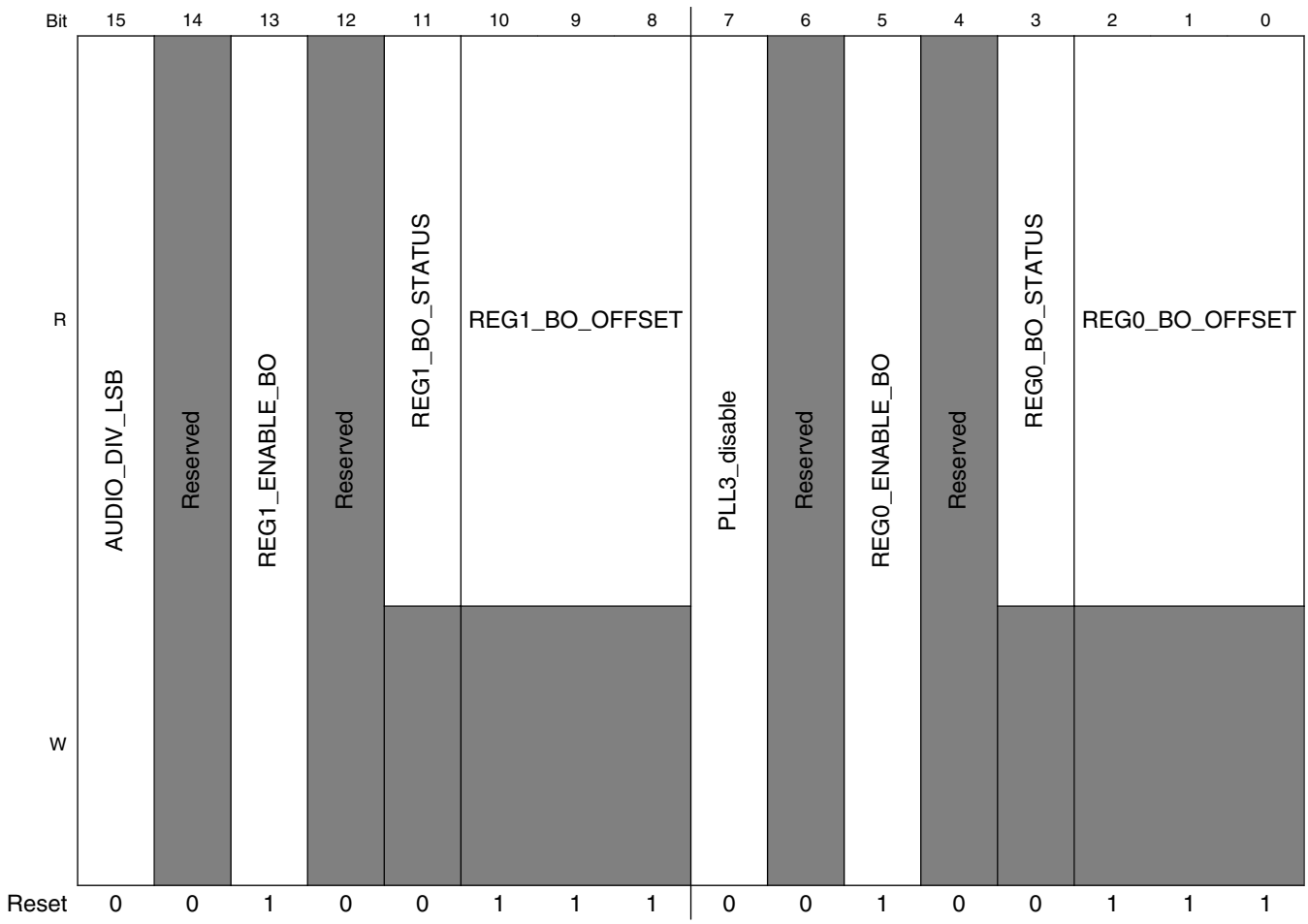
This register defines the control for miscellaneous PMU Analog blocks.

NOTE

This register is shared with CCM.

Address: 20C_8000h base + 170h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	VIDEO_DIV		REG2_STEP_TIME		REG1_STEP_TIME		REG0_STEP_TIME		AUDIO_DIV_MSB	REG2_OK	REG2_ENABLE_BO	Reserved	REG2_BO_STATUS	REG2_BO_OFFSET		
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1



PMU_MISC2*n* field descriptions

Field	Description
31–30 VIDEO_DIV	Post-divider for video. The output clock of the video PLL should be gated prior to changing this divider to prevent glitches. This divider is feed by PLL_VIDEOn[POST_DIV_SELECT] to achieve division ratios of /1, /2, /4, /8, and /16. **NOTE:** Not related to PMU. See Clock Controller Module (CCM) for more information. 00 divide by 1 (Default) 01 divide by 2 10 divide by 1 11 divide by 4
29–28 REG2_STEP_TIME	Number of clock periods (24MHz clock). 00 **64_CLOCKS** — 64 01 **128_CLOCKS** — 128 10 **256_CLOCKS** — 256 11 **512_CLOCKS** — 512
27–26 REG1_STEP_TIME	Number of clock periods (24MHz clock). 00 **64_CLOCKS** — 64 01 **128_CLOCKS** — 128

Table continues on the next page...

PMU_MISC2*n* field descriptions (continued)

Field	Description
	10 **256_CLOCKS** — 256 11 **512_CLOCKS** — 512
25–24 REG0_STEP_TIME	Number of clock periods (24MHz clock). 00 **64_CLOCKS** — 64 01 **128_CLOCKS** — 128 10 **256_CLOCKS** — 256 11 **512_CLOCKS** — 512
23 AUDIO_DIV_MSB	MSB of Post-divider for Audio PLL. The output clock of the video PLL should be gated prior to changing this divider to prevent glitches. This divider is feed by PLL_AUDIOn[POST_DIV_SELECT] to achieve division ratios of /1, /2, /4, /8, and /16. **NOTE:** MSB bit value pertains to the first bit, please program the LSB bit (bit 15) as well to change divider value **NOTE:** Not related to PMU. See Clock Controller Module (CCM) for more information. 00 divide by 1 (Default) 01 divide by 2 10 divide by 1 11 divide by 4
22 REG2_OK	Signals that the voltage is above the brownout level for the SOC supply. 1 = regulator output > brownout_target
21 REG2_ENABLE_BO	Enables the brownout detection.
20 -	This field is reserved.
19 REG2_BO_STATUS	Reg2 brownout status bit.
18–16 REG2_BO_OFFSET	This field defines the brown out voltage offset for the xPU power domain. IRQ_DIG_BO is also asserted. Single-bit increments reflect 25mV brownout voltage steps. The reset brown-offset is 175mV below the programmed target code. Brownout target = OUTPUT_TRG - BO_OFFSET. Some steps may be irrelevant because of input supply limitations or load operation. 100 Brownout offset = 0.100V 111 Brownout offset = 0.175V
15 AUDIO_DIV_LSB	LSB of Post-divider for Audio PLL. The output clock of the video PLL should be gated prior to changing this divider to prevent glitches. This divider is feed by PLL_AUDIOn[POST_DIV_SELECT] to achieve division ratios of /1, /2, /4, /8, and /16. **NOTE:** LSB bit value pertains to the last bit, please program the MSB bit (bit 23) as well, to change divider value **NOTE:** Not related to PMU. See Clock Controller Module (CCM) for more information. 00 divide by 1 (Default) 01 divide by 2 10 divide by 1 11 divide by 4

Table continues on the next page...

PMU_MISC2*n* field descriptions (continued)

Field	Description
14 -	This field is reserved. Reserved
13 REG1_ENABLE_BO	Enables the brownout detection.
12 -	This field is reserved.
11 REG1_BO_STATUS	Reg1 brownout status bit. 1 Brownout, supply is below target minus brownout offset.
10–8 REG1_BO_OFFSET	This field defines the brown out voltage offset for the xPU power domain. IRQ_DIG_BO is also asserted. Single-bit increments reflect 25mV brownout voltage steps. The reset brown-offset is 175mV below the programmed target code. Brownout target = OUTPUT_TRG - BO_OFFSET. Some steps may be irrelevant because of input supply limitations or load operation. 100 Brownout offset = 0.100V 111 Brownout offset = 0.175V
7 PLL3_disable	Default value of "0". Should be set to "1" to turn off the USB-PLL(PLL3) in run mode. **NOTE:** Not related to PMU. See Clock Controller Module (CCM) for more information.
6 -	This field is reserved.
5 REG0_ENABLE_BO	Enables the brownout detection.
4 -	This field is reserved.
3 REG0_BO_STATUS	Reg0 brownout status bit. 1 Brownout, supply is below target minus brownout offset.
REG0_BO_OFFSET	This field defines the brown out voltage offset for the CORE power domain. IRQ_DIG_BO is also asserted. Single-bit increments reflect 25mV brownout voltage steps. Some steps may be irrelevant because of input supply limitations or load operation. 100 Brownout offset = 0.100V 111 Brownout offset = 0.175V

Chapter 51
Pulse Width Modulation (PWM)

51.1 Overview

The Pulse Width Modulation (PWM) has a 16-bit counter, and is optimized to generate sound from stored sample audio images and it can also generate tones. It uses 16-bit resolution and a 4 x 16 data FIFO.

This section presents an overview of the PWM. A block diagram of the PWM module is shown in the figure below.

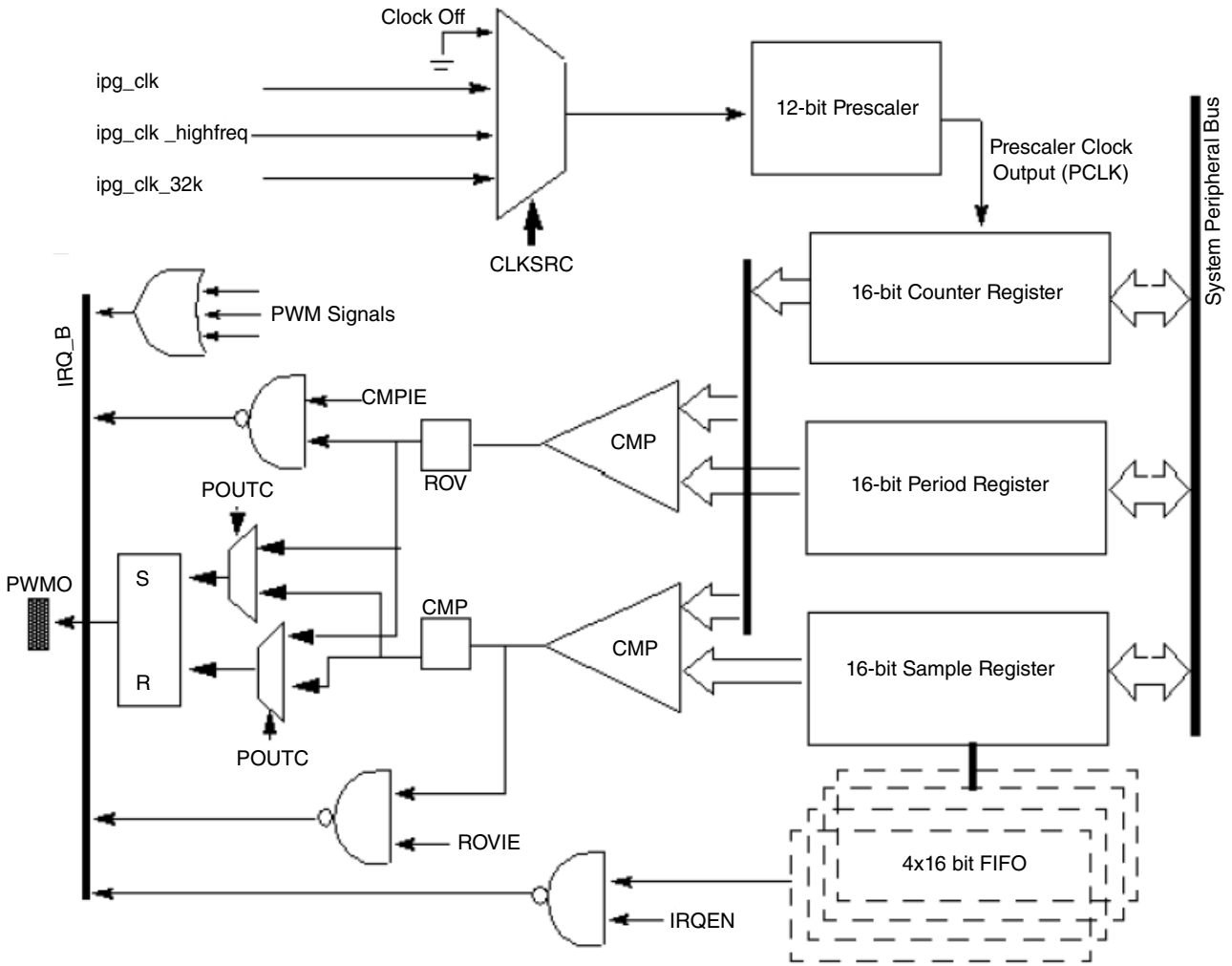


Figure 51-1. Pulse-Width Modulator Block Diagram

The following features characterize the PWM:

- 16-bit up-counter with clock source selection
- 4 x 16 FIFO to minimize interrupt overhead
- 12-bit prescaler for division of clock
- Sound and melody generation
- Active high or active low configured output
- Can be programmed to be active in low-power mode
- Can be programmed to be active in debug mode
- Interrupts at compare and rollover

51.2 External Signals

The PWM follows IP Bus protocol when interfacing with the processor core. PWM does not have any interface signals with any other block inside the chip except for clock and reset inputs from the Clock Control Module (CCM), System Reset Controller (SRC), and interrupt signals to the processor interrupt handler. There is a single output signal.

The following table outlines the external signals.

Table 51-1. PWM External Signals

Signal	Description	Pad	Mode	Direction
PWM1_OUT	This is the PWM1 functional output of the PWM. A modulated signal of the block is observed at this pin. It can be viewed as a clock signal whose period and duty cycle can be varied with different settings of the cycle of 50%.	DISP0_DAT8	ALT2	O
		GPIO_9	ALT4	
		SD1_DAT3	ALT3	
PWM2_OUT	This is the PWM2 functional output of the PWM. A modulated signal of the block is observed at this pin. It can be viewed as a clock signal whose period and duty cycle can be varied with different settings of the cycle of 50%.	DISP0_DAT9	ALT2	O
		GPIO_1	ALT4	
		SD1_DAT2	ALT3	
PWM3_OUT	This is the PWM3 functional output of the PWM. A modulated signal of the block is observed at this pin. It can be viewed as a clock signal whose period and duty cycle can be varied with different settings of the cycle of 50%.	SD1_DAT1	ALT2	O
		SD4_DAT1	ALT2	
PWM4_OUT	This is the PWM4 functional output of the PWM. A modulated signal of the block is observed at this pin. It can be viewed as a clock signal whose period and duty cycle can be varied with different settings of the cycle of 50%.	SD1_CMD	ALT2	O
		SD4_DAT2	ALT2	

51.3 Clocks

The table found here describes the clock sources for PWM.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 51-2. PWM Clocks

Clock name	Clock Root	Description
ipg_clk	ipg_clk_root	Peripheral clock
ipg_clk_32k	ckil_sync_clk_root	low-frequency reference clock (32kHz)
ipg_clk_highfreq	perclk_clk_root	high-frequency reference clock
ipg_clk_s	ipg_clk_root	Peripheral access clock

The clock that feeds the prescaler can be selected from:

- High-frequency reference clock (ipg_clk_highfreq) pat_ref or CKIH

  This is a high frequency clock, provided by the Clock Control Module (CCM). This clock should be on in the low power mode when the ipg_clk is turned off. Thus, the PWM can be run on this clock in the low power mode.

- Low-frequency reference clock (ipg_clk_32k, CKIL)

  This is the 32 KHz low reference clock which is provided by the CCM. This clock should be on in the low power mode when ipg_clk is turned off. Thus, PWM can be run on this clock in the low power mode.

- Peripheral clock (ipg_clk)

  This clock should be on in normal operations. In low power mode, it can be switched off.

- Peripheral access clock (ipg_clk_s)

  This clock is used for register read/write.

The clock input source is determined by the PWM control register field PWM_CR[CLKSRC]. The CLKSRC value should only be changed when the PWM is disabled.

A change in the value of the PRESCALER field of the control register is immediately reflected on its output clock frequency.

51.4 Functional Description

The following sections detail the PWM operation and function.

51.4.1 Operation

The output of the PWM is a toggling signal whose frequency and duty cycle can be modulated by programming the appropriate registers. It has a 16-bit up counter which counts from 0x0000 until the counter value equals the PWM_PR + 1. After this match occurs the counter is reset to 0x0000.

At the beginning of a count period cycle, the PWMO pin is set to one (default) and the counter begins counting up from 0x0000. The sample value in the sample FIFO is compared on each count of prescaler clock. When the sample and count values match, the PWMO signal is cleared to zero (default). The counter continues counting until the period match occurs and subsequently another period cycle begins.

When the PWM is enabled, the counter starts running and generates an output with the reset values in the period and sample registers. It is recommended that the programming of these registers be done before PWM is enabled.

A hardware reset results in all the PWM count and sample registers being cleared and the FIFO being flushed. The control register shows that FIFO is empty and it can be written into, and the PWM is disabled. A software reset has the same results, however the state of the STOPEN, DOZEN, WAITEN, and DBGEN bits in the control register are not affected. Software reset can be asserted even when the PWM is in disabled state.

51.4.1.1 FIFO

Digital sample values can be loaded into the pulse-width modulator as 16-bit words. The endianess can be changed using the BCTR and HCTR bits of the control register. A 4-word (16-bit) FIFO minimizes interrupt overhead. A maskable interrupt is generated when the number of data words fall below the water level set by the FWM field in the control register.

A write to the PWM_SAR sample register results in the value being stored into the FIFO if it is not full. A write when the FIFO is full sets FWE (FIFO write error) bit in the status register and the FIFO contents remain unchanged. The FIFO can be written at any time, but can be read only when the PWM is enabled. The PWM_SR[FIFOAV] field shows how many data words are currently contained in the FIFO and whether or not it can be written into.

A read on the sample register yields the current FIFO value that is being used, or will be used, by the PWM for generation on the output signal. Therefore, a write and a subsequent read on the sample register may result in different values being obtained.

51.4.1.2 Rollover and Compare Event

The counter is reset to 0x0000 after its value equals the PWM_PR[PERIOD] + 1 and resumes counting thereafter. This event is referred to as a rollover. For example, if PWM_PR[PERIOD] = 0x0000, the counter is reset when it equals 0x0001. When PWM_PR[PERIOD] = 0xFFFF or 0xFFFE, the counter is reset when it equals 0xFFFF. For more information, see the PWM Period Register (PWM_PR) description.

During a rollover event the output is either set (default), reset or has no effect according to the programming of the POUTC field in the control register. This event can also generate an interrupt if the respective interrupt enable bit is set in the control register.

When the counter value reaches the sample value, the output of the PWM is reset (default), set or has no effect according to the programming of the POUTC field of control register. This event is referred to as a compare event. This event can also generate an interrupt if the respective interrupt enable bit is set in the control register.

If the rollover event sets the PWM output signal, the compare event will reset it and vice versa for a particular programming configuration of POUTC field.

51.4.1.3 Low Power Mode Behavior

In low power mode, if the clock from the selected clock source is available, the PWM counter continues to run and an output is produced, depending on whether the control bit for that mode is set or not. In the absence of the clock itself, or if the corresponding low power bit in the control register is 0, the counter is reset and resumes counting when it exits the low power mode.

51.4.1.4 Debug Mode Behavior

In debug mode, PWM has the option of continuing to run or be halted. If the DBGEN bit is not set in the PWM_PWMCR, the PWM is halted. If the DBGEN bit is set, then the PWM will continue to run in the debug mode.

51.5 Enable Sequence for the PWM

The sequence found here should be used to enable the PWM.

1. Configure the desired settings for the PWM Control Register (PWMx_PWMCR) while keeping the PWM disabled (PWMx_PWMCR[0]=0).
2. Enable the desired interrupts in the PWM Interrupt Register (PWMx_PWMIR).
3. One to three initial samples may be written to the PWM Sample Register (PWMx_PWMSAR). The initial sample values will be loaded into the PWM FIFO even if the PWM is not yet enabled. Do not write a 4th sample because the FIFO will become full and trigger a FIFO Write Error (FWE). This error will prevent the PWM from starting once it is enabled.
4. Check the FIFO Write Error status bit (FWE), the Compare status bit (CMP) and the Roll-over status bit (ROV) in the PWM Status Register (PWMx_PWMSR) to make sure they are all zero. Any non-zero status bits should be cleared by writing a 1 to them.
5. Write the desired period to the PWM Period Register (PWMx_PWMPR).
6. Enable the PWM by writing a 1 to the PWM Enable bit, PWMx_PWMCR[0], while maintaining the other register bits in their previously configured state.

51.6 Disable Sequence for the PWM

The PWM can be disabled at any time by clearing the PWM enable bit, PWMx_PWMCR[0] to 0.

Any data remaining in the FIFO will not be produced at the PWM output after the PWM has been disabled and will remain in the FIFO until the PWM is enabled again. A software reset (setting PWMx_PWMCR[3] to 1) or a hardware reset will clear the FIFO and any remaining data will be lost.

51.7 PWM Memory Map/Register Definition

The PWM includes six user-accessible 32-bit registers.

PWM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
208_0000	PWM Control Register (PWM1_PWMCR)	32	R/W	0000_0000h	51.7.1/4467
208_0004	PWM Status Register (PWM1_PWMSR)	32	w1c	0000_0008h	51.7.2/4469
208_0008	PWM Interrupt Register (PWM1_PWMIR)	32	R/W	0000_0000h	51.7.3/4470

Table continues on the next page...

PWM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
208_000C	PWM Sample Register (PWM1_PWMSAR)	32	R/W	0000_0000h	51.7.4/4471
208_0010	PWM Period Register (PWM1_PWMPR)	32	R/W	0000_FFFEh	51.7.5/4472
208_0014	PWM Counter Register (PWM1_PWMCNR)	32	R	0000_0000h	51.7.6/4473
208_4000	PWM Control Register (PWM2_PWMCR)	32	R/W	0000_0000h	51.7.1/4467
208_4004	PWM Status Register (PWM2_PWMSR)	32	w1c	0000_0008h	51.7.2/4469
208_4008	PWM Interrupt Register (PWM2_PWMIR)	32	R/W	0000_0000h	51.7.3/4470
208_400C	PWM Sample Register (PWM2_PWMSAR)	32	R/W	0000_0000h	51.7.4/4471
208_4010	PWM Period Register (PWM2_PWMPR)	32	R/W	0000_FFFEh	51.7.5/4472
208_4014	PWM Counter Register (PWM2_PWMCNR)	32	R	0000_0000h	51.7.6/4473
208_8000	PWM Control Register (PWM3_PWMCR)	32	R/W	0000_0000h	51.7.1/4467
208_8004	PWM Status Register (PWM3_PWMSR)	32	w1c	0000_0008h	51.7.2/4469
208_8008	PWM Interrupt Register (PWM3_PWMIR)	32	R/W	0000_0000h	51.7.3/4470
208_800C	PWM Sample Register (PWM3_PWMSAR)	32	R/W	0000_0000h	51.7.4/4471
208_8010	PWM Period Register (PWM3_PWMPR)	32	R/W	0000_FFFEh	51.7.5/4472
208_8014	PWM Counter Register (PWM3_PWMCNR)	32	R	0000_0000h	51.7.6/4473
208_C000	PWM Control Register (PWM4_PWMCR)	32	R/W	0000_0000h	51.7.1/4467
208_C004	PWM Status Register (PWM4_PWMSR)	32	w1c	0000_0008h	51.7.2/4469
208_C008	PWM Interrupt Register (PWM4_PWMIR)	32	R/W	0000_0000h	51.7.3/4470
208_C00C	PWM Sample Register (PWM4_PWMSAR)	32	R/W	0000_0000h	51.7.4/4471
208_C010	PWM Period Register (PWM4_PWMPR)	32	R/W	0000_FFFEh	51.7.5/4472
208_C014	PWM Counter Register (PWM4_PWMCNR)	32	R	0000_0000h	51.7.6/4473

51.7.1 PWM Control Register (PWMx_PWMCR)

The PWM control register (PWM_PWMCR) is used to configure the operating settings of the PWM. It contains the prescaler for the clock division.

Address: Base address + 0h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0				FWM		STOPEN	DOZEN	WAITEN	DBGEN	BCTR	HCTR	POUTC		CLKSRC	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PRESCALER												SWR	REPEAT		EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMx_PWMCR field descriptions

Field	Description
31–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–26 FWM	FIFO Water Mark. These bits are used to set the data level at which the FIFO empty flag will be set and the corresponding interrupt generated 00 FIFO empty flag is set when there are more than or equal to 1 empty slots in FIFO 01 FIFO empty flag is set when there are more than or equal to 2 empty slots in FIFO 10 FIFO empty flag is set when there are more than or equal to 3 empty slots in FIFO 11 FIFO empty flag is set when there are more than or equal to 4 empty slots in FIFO
25 STOPEN	Stop Mode Enable. This bit keeps the PWM functional while in stop mode. When this bit is cleared, the input clock is gated off in stop mode. This bit is not affected by software reset. It is cleared by hardware reset. 0 Inactive in stop mode 1 Active in stop mode
24 DOZEN	Doze Mode Enable. This bit keeps the PWM functional in doze mode. When this bit is cleared, the input clock is gated off in doze mode. This bit is not affected by software reset. It is cleared by hardware reset. 0 Inactive in doze mode 1 Active in doze mode
23 WAITEN	Wait Mode Enable. This bit keeps the PWM functional in wait mode. When this bit is cleared, the input clock is gated off in wait mode. This bit is not affected by software reset. It is cleared by hardware reset.

Table continues on the next page...

PWMx_PWMCR field descriptions (continued)

Field	Description
	0 Inactive in wait mode 1 Active in wait mode
22 DBGEN	Debug Mode Enable. This bit keeps the PWM functional in debug mode. When this bit is cleared, the input clock is gated off in debug mode. This bit is not affected by software reset. It is cleared by hardware reset. 0 Inactive in debug mode 1 Active in debug mode
21 BCTR	Byte Data Swap Control. This bit determines the byte ordering of the 16-bit data when it goes into the FIFO from the sample register. 0 byte ordering remains the same 1 byte ordering is reversed
20 HCTR	Half-word Data Swap Control. This bit determines which half word data from the 32-bit IP Bus interface is written into the lower 16 bits of the sample register. 0 Half word swapping does not take place 1 Half words from write data bus are swapped
19–18 POUTC	PWM Output Configuration. This bit field determines the mode of PWM output on the output pin. 00 Output pin is set at rollover and cleared at comparison 01 Output pin is cleared at rollover and set at comparison 10 PWM output is disconnected 11 PWM output is disconnected
17–16 CLKSRC	Select Clock Source. These bits determine which clock input will be selected for running the counter. After reset the system functional clock is selected. The input clock can also be turned off if these bits are set to 00. This field value should only be changed when the PWM is disabled 00 Clock is off 01 ipg_clk 10 ipg_clk_highfreq 11 ipg_clk_32k
15–4 PRESCALER	Counter Clock Prescaler Value. This bit field determines the value by which the clock will be divided before it goes to the counter. 0x000 Divide by 1 0x001 Divide by 2 0xfff Divide by 4096
3 SWR	Software Reset. PWM is reset when this bit is set to 1. It is a self clearing bit. A write 1 to this bit is a single wait state write cycle. When the block is in reset state this bit is set and is cleared when the reset procedure is over. Setting this bit resets all the registers to their reset values except for the STOPEN, DOZEN, WAITEN, and DBGEN bits in this control register. 0 PWM is out of reset 1 PWM is undergoing reset
2–1 REPEAT	Sample Repeat. This bit field determines the number of times each sample from the FIFO is to be used. 00 Use each sample once 01 Use each sample twice 10 Use each sample four times 11 Use each sample eight times

Table continues on the next page...

PWMx_PWMCR field descriptions (continued)

Field	Description
0 EN	PWM Enable. This bit enables the PWM. If this bit is not enabled, the clock prescaler and the counter is reset. When the PWM is enabled, it begins a new period, the output pin is set to start a new period while the prescaler and counter are released and counting begins. To make the PWM work with softreset and disable/enable, users can do software reset by seting the SWR bit, wait software reset done, configure the registers, and then enable the PWM by setting this bit to "1" Users can also disable/enable the PWM if PWM would like to be stopped and resumed with same registers configurations . 0 PWM disabled 1 PWM enabled

51.7.2 PWM Status Register (PWMx_PWMSR)

The PWM status register (PWM_PWMSR) contains seven bits which display the state of the FIFO and the occurrence of rollover and compare events. The FIFOAV bit is read-only but the other four bits can be cleared by writing 1 to them. The FE, ROV, and CMP bits are associated with FIFO-Empty, Roll-over, and Compare interrupts, respectively.

Address: Base address + 4h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0									FWE	CMP	ROV	FE	FIFOAV		
W										w1c	w1c	w1c	w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

PWMx_PWMSR field descriptions

Field	Description
31–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 FWE	FIFO Write Error Status. This bit shows that an attempt has been made to write FIFO when it is full. 0 FIFO write error not occurred 1 FIFO write error occurred
5 CMP	Compare Status. This bit shows that a compare event has occurred. 0 Compare event not occurred 1 Compare event occurred

Table continues on the next page...

PWMx_PWMSR field descriptions (continued)

Field	Description
4 ROV	Roll-over Status. This bit shows that a roll-over event has occurred. 0 Roll-over event not occurred 1 Roll-over event occurred
3 FE	FIFO Empty Status Bit. This bit indicates the FIFO data level in comparison to the water level set by FWM field in the control register. 0 Data level is above water mark 1 When the data level falls below the mark set by FWM field
FIFOAV	FIFO Available. These read-only bits indicate the data level remaining in the FIFO. An attempted write to these bits will not affect their value and no transfer error is generated. 000 No data available 001 1 word of data in FIFO 010 2 words of data in FIFO 011 3 words of data in FIFO 100 4 words of data in FIFO 101 unused 110 unused 111 unused

51.7.3 PWM Interrupt Register (PWMx_PWMIR)

The PWM Interrupt register (PWM_PWMIR) contains three bits which control the generation of the compare, rollover and FIFO empty interrupts.

Address: Base address + 8h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0													CIE	RIE	FIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMx_PWMIR field descriptions

Field	Description
31–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 CIE	Compare Interrupt Enable. This bit controls the generation of the Compare interrupt.

Table continues on the next page...

PWMx_PWMIR field descriptions (continued)

Field	Description
	0 Compare Interrupt not enabled 1 Compare Interrupt enabled
1 RIE	Roll-over Interrupt Enable. This bit controls the generation of the Rollover interrupt. 0 Roll-over interrupt not enabled 1 Roll-over Interrupt enabled
0 FIE	FIFO Empty Interrupt Enable. This bit controls the generation of the FIFO Empty interrupt. 0 FIFO Empty interrupt disabled 1 FIFO Empty interrupt enabled

51.7.4 PWM Sample Register (PWMx_PWMSAR)

The PWM sample register (PWM_PWMSAR) is the input to the FIFO. 16-bit words are loaded into the FIFO. The FIFO can be written at any time, but can be read only when the PWM is enabled. The PWM will run at the last set duty-cycle setting if all the values of the FIFO has been utilized, until the FIFO is reloaded or the PWM is disabled. When a new value is written, the duty cycle changes after the current period is over.

A value of zero in the sample register will result in the PWMO output signal always being low/high (POUTC = 00 it will be low and POUTC = 01 it will be high), and no output waveform will be produced. If the value in this register is higher than the PERIOD + 1, the output will never be set/reset depending on POUTC value.

Address: Base address + Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																SAMPLE															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMx_PWMSAR field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SAMPLE	Sample Value. This is the input to the 4x16 FIFO. The value in this register denotes the value of the sample being currently used.

51.7.5 PWM Period Register (PWMx_PWMPR)

The PWM period register (PWM_PWMPR) determines the period of the PWM output signal. After the counter value matches PERIOD + 1, the counter is reset to start another period.

PWMO (Hz) = PCLK(Hz) / (period +2)

A value of zero in the PWM_PWMPR will result in a period of two clock cycles for the output signal. Writing 0xFFFF to this register will achieve the same result as writing 0xFFFE.

A change in the period value due to a write in PWM_PWMPR results in the counter being reset to zero and the start of a new count period.

NOTE

Settings PWM_PWMPR to 0xFFFF when PWMx_PWMCR REPEAT bits are set to non-zero values is not allowed.

Address: Base address + 10h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																PERIOD															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

PWMx_PWMPR field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
PERIOD	Period Value. These bits determine the Period of the count cycle. The counter counts up to [Period Value] +1 and is then reset to 0x0000.

51.7.6 PWM Counter Register (PWMx_PWMCNR)

The read-only pulse-width modulator counter register (PWM_PWMCNR) contains the current count value and can be read at any time without disturbing the counter.

Address: Base address + 14h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMx_PWMCNR field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	Counter Value. These bits are the counter register value and denotes the current count state the counter register is in.

Chapter 52
ROM Controller with Patch (ROMC)

52.1 Overview

The Read Only Memory Controller with ROM Patch (ROMC) acts as an interface between the ARM advanced high-performance bus (AHB - Lite) and the Read Only Memory. The ROMC consists of a ROM Controller and a ROM Patch. The ROM Patch is used to either patch code routines or fix data tables in the ROM area. There is an IP Bus interface to access the ROM Patch Registers and. The figure below depicts the main functional sub-blocks of the ROMC.

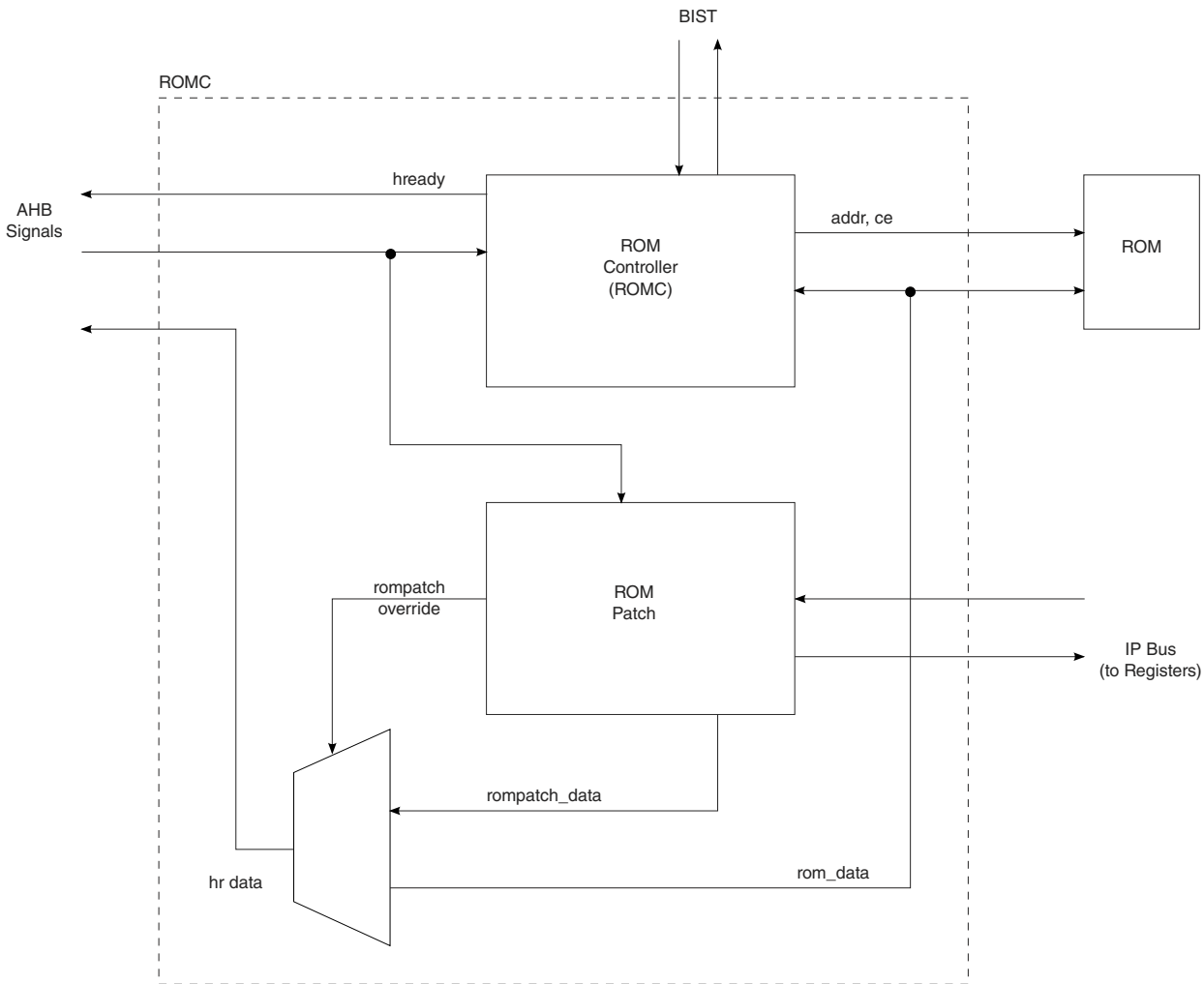


Figure 52-1. ROMC Block Diagram

52.1.1 Features

- Supports ROM size ranges from 16 Kbyte up to 4 Mbyte with increments of 1 Kbyte
- Supports opcode patching for a maximum of 16 different addresses in 4 Mbytes of ROM space
- Supports one-word data fixes for a max of 8 memory locations in 4 Mbytes of ROM space
- Supports patching of the Reset Vector (at 0x0000_0000) to allow external booting

52.1.2 Modes of Operation

There are two modes of operation: normal mode and BIST mode.

In normal mode (ipt_bist_en = 0), the ROMC ensures correct reads from the ROM, assuming the memory complies with the characteristics and requirements for which the ROMC was designed.

52.1.2.1 Low Power Mode

There are two clock enables that are used to switch off parts of the ROMC logic when inactive. The first clock enable is used to disable the ROM Controller when the master connected to the AHB interface is not initiating a read to the ROM. The second clock enable is used to disable the registers used to program the ROM patch feature when the registers are not being accessed.

52.2 Clocks

The table found here describes the clock sources for ROMCP.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 52-1. ROMCP Clocks

Clock name	Clock Root	Description
hclk	ahb_clk_root	System / bus clock
hclk_reg	ipg_clk_root	System access clock
96krom_CLK	ahb_clk_root	ROM clock

52.3 Memory Map

52.3.1 ROM Memory Map in detail

The ROMC supports ROM sizes with a range of 16 Kbyte to 4 Mbyte with an increment of 1 Kbyte. The 16 Kbyte lower limit was chosen because the minimum size of security code on an ARM platform is approximately 16 Kbyte of code, which is only accessible in supervisor mode. Note that it is the MMU that controls whether any region of memory is secure.

The exception vectors must be secured as well, and must be put in the same area as the security code. Since they must reside at address 0x0000_0000, the entire 16 Kbyte of ROM which can only be accessible in supervisor mode is located at the very beginning of the platform memory map.

If the user chooses not to use the security code, a memory size smaller than 16 Kbyte can be connected to the platform (minimum of 1 Kbyte). The MMU can be programmed to allow any kind of access into this memory. However, if the ROM size is less than 16 Kbyte, memory aliasing will occur for all invalid addresses greater than the memory size but within the 16 Kbyte of space.

For ROM sizes bigger than 16 Kbyte, the rest of its physical size resides at the address starting at 0x0040_4000 (4 M+16 Kbyte) going up to [0x0040_4000 + (mem. size - 16Kbyte)]. For examples on power-of-two memories please see Figure 1.

52.4 Functional Description

This section is divided up into the ROM Controller Functional Description and the ROMC functional description.

52.4.1 ROM Controller (ROMC) Functional Description

52.4.1.1 Functionality overview

The ROMC serves two main functions. First, as an interface between the AHB-Lite bus on an ARM platform and the ROM. Second, it drives and receives several signals for the BIST engine. In normal mode of operation, the ROMC monitors the AHB-Lite for memory access requests and performs the memory operation to the ROM.

The ROMC includes the option to wait state all accesses from either the ARM or non-ARM masters to ROM in the event that timing requirements will not allow single hclk clock cycle reads. If a wait state is required, the static inputs rom_wait_arm or rom_wait_alt_mstr can be set to 1 and accesses will take two hclk clock cycles. If wait states are not required, rom_wait_arm or rom_wait_alt_mstr can be set to 0 and accesses will take one hclk clock cycle to complete.

52.4.2 ROMC Functional Description

52.4.2.1 ROMC Disabling

All the bits in the ROMC_ROMPATCHENL register are cleared on Reset, disabling all the address comparators. Once the comparators have been enabled, the ROMC functions of data fixing and opcode patching can be quickly disabled by setting the DIS bit in the ROMC_ROMPATCHCNTL register. This bit is used to enable secure operations in which patching functions need to be disabled. This bit is cleared on Reset.

52.4.2.2 ROMC Event Priority

The ROMC has a total of 16 address comparators. The first 8 (0 through 7) comparators can be programmed for the data fixing function (through the 8 data fix enable bits in the ROMC_ROMPATCHCNTL register) while the rest are for opcode patching by default. This allows for potential multiple matching events involving both data fixing and opcode patch types. In these cases the ROMC assigns the highest priority to a data fixing event.

For example, if the ROMC is set up to data fix a certain address with comparator 4 and also opcode patch the same address with comparator 7, it will let comparator 4 have higher priority in indicating a match, and data from ROMC_ROMPATCHD4 will be put on the rompatch_romc_hrdata bus as the override value.

If multiple address matches of the same type level occur concurrently, then the ROMC will choose the source number based on the one with the highest source number. For example, the ROMC is setup to data fix the same location with address comparators 4 and 7, then address comparator 7 will have higher priority in indicating a match, and the value from ROMC_ROMPATCHD7 will be put on the rompatch_romc_hrdata bus as the orverride value. The same priority applies for an opcode patch event, except the override data is in the form of an SWI instruction with the comment field set to the source number with the highest priority.

52.4.2.3 Data Fixing

The data fixing feature allows ROM data to be updated by direct replacement when it is being read. This data usually originates from data tables, but can include ARM instructions. To enable data fixing on a certain address, this address value is written in to one of the first eight (0 through 7) of ROMC_ROMPATCHAxx registers and the same numbered bit set in the ROMC_ROMPATCHENL and ROMC_ROMPATCHCNTL registers. The data to be used for replacement is placed in the corresponding ROMC_ROMPATCHDxx.

The ROMC looks for a read access to ROM (either code fetch or data load) by snooping the AHB interface for read transactions. The address is compared with the values stored in the ROMC_ROMPATCHAxx[22:2] registers. If a match occurs from one of the comparators, the ROMC places the value in the corresponding ROMC_ROMPATCHDxx register on the read data bus by overriding the read data coming from the actual ROM (see the mux in Figure 52-1). The value on the read data bus is maintained until hready is asserted to terminate the access. In data fixing, the entire word is replaced so if a byte or half-word access occurs on a "data fix" location, the entire data word is replaced. The word being replaced is word aligned. (The two LSBs of the matching ROMC_ROMPATCHAxx are ignored in the data fix operation.)

52.4.2.4 Opcode Patching

The opcode patch feature provides the ARM core a mechanism to fetch updated versions of code routines that were originally programmed in ROM. This patching mechanism makes use of the SWI (software interrupt instruction) and a table of function pointers residing in writable memory. The opcode being patched is replaced with a SWI instruction by the ROMC. Subsequent processing of the SWI reads from a function pointers table to obtain the address of the replacement code. Execution resumes with this code patch.

To enable opcode patching of a certain address, this address value is written into one of the ROMPATCHAxx registers and the corresponding bit set in the ROMPATCHENL to enable the associated comparator. The register's LSB (ROMC_ROMPATCHxx[0]) should be set if THUMB mode patching is in effect for this address. The ROMC identifies a ROM read access by snooping the AHB interface. The address is compared with the values stored in the ROMC_ROMPATCHAxx[22:2] registers. If a match occurs from one of the comparators, the ROMC generates the opcode of a software interrupt (SWI) instruction with the comment field containing the number of the matching address comparator. This opcode and comment is placed on the read data bus until hready is asserted by the ROM controller to terminate the read access.

The type of SWI generated, (that is, either ARM or THUMB), is determined by the LSB of the ROMC_ROMPATCHAxx register associated with the opcode patch. This bit is cleared for ARM mode (32 bits). The ROMC generates a 32-bit SWI (opcode field is 0xEF, occupying bits [31:24] of the word), with the least significant 5 bits of the 24-bit comment field (bits [23:0]) containing the number of the matching address comparator. The rest of the comment field is filled with zeros. This means that the ROMC will use 16 of the 16777216 possible software interrupts. The ROMC overrides the read data from the ROM.

If the LSB of the matching ROMC_ROMPATCHAxx register is set, the opcode patch is in THUMB mode (16 bits or half word). The ROMC generates a 16-bit SWI instruction (opcode field is 0xDF, occupying bits [15:8] of the half word) with the least significant 5 bits of the 8-bit comment field containing with the source number of the address comparator. The rest of the comments field is filled with zeros. This means that the ROMC will use 16 of the 256 possible software interrupts. The ROMC puts this 16 bit SWI instruction value on the proper half of the rompatch_romc_hrdata bus. The other half is zeroed out. Which half of the bus contains the SWI opcode and comment depends on the mode (Big Endian or Little Endian) and the bit 1 of the matching ROMC_ROMPATCHAxx register. In Little Endian mode, the lower half is bits {15:0] and the upper half is bits {31:16]. The order is reversed in Big Endian mode.

In Little Endian mode (bigend signal negated), if bit 1 of the matching ROMC_ROMPATCHAxx is cleared (lower half word selected) then the SWI instruction is put on the lower 16 bits of the read data bus and the upper 16 bits are zeroed out. Only the lower 16 bits of the read data bus is overwritten by the ROMC data. If ROMC_ROMPATCHAxx[1] is set (upper half word selected), the SWI instruction is put on the upper 16 bits of the read data bus and the lower 16 bits are zeroed out. Only the upper 16 bits of the read data bus is overwritten.

In Big Endian mode (bigend asserted), if bit 1 of the matching ROMC_ROMPATCHAxx is cleared (lower half word selected) then the SWI instruction is put on the upper 16 bits of the read data bus while the lower 16 bits are zeroed out. Only the upper 16 bits of the read data bus is overwritten. If ROMC_ROMPATCHAxx[1] is set (upper word selected), the SWI instruction is put on the lower 16 bits and the upper 16 bits are zeroed out. Only the lower 16 bits of the read data bus is overwritten.

The eventual execution of the SWI causes the ARM to save the CPSR in SPSR_SVC, the address of the next instruction after the SWI in R14_SVC, enter Supervisor mode, and fetch the SWI vector at 0x8, which then takes it to a handler for further processing as described in the next section.

52.4.2.4.1 Typical Software Response to Opcode Patch

When the SWI handler executes it needs to determine whether the SWI was generated by the ROMC. This is done by loading the SWI instruction and extracting its comment field. The state of the ARM core (ARM or THUMB) when the SWI was executed dictates whether to load the instruction word (ARM) or half word (THUMB). This state information can be determined by testing the T bit (bit 5) of the SPSR. If it's set, the execution was in THUMB mode.

By convention, if the comment field of the SWI is greater than 16, the software interrupt was initiated by software (i.e. an operating system call), and a branch is taken to the appropriate handler routine for further processing. If the comment field is less than 16, the SWI was generated by the ROMC performing a code patch operation. In this case, the software then reads from a table of function pointers, using the value in the SWI comment field as the index into the table. The value that is read is the address of the code patch. This value is loaded into the PC to begin the execution of the code patch. The following code segment illustrates a typical handling of the SWI.

```
stmfd        sp!, {r0-r1,lr}       @ push register onto SWI stack
mrs         r0, spsr        @ get saved status register
tst          r0, #0x20     @ check if call was in THUMB mode
ldrneh      r0, [lr,#-2]       @ yes: load opcode half-word and
bicne       r0, r0, #0xff00   @ yes: extract THUMB comment
ldreq       r0, [lr,#-4]       @ no: load opcode word and
biceq       r0, r0, #0xff000000 @ no: extract ARM comment
                    @ now r0 has comment field
cmp        r0, #16      @ compare to 16 (maximum for ROMC)
ldrlt         lr, =rompatch_tbl_ptr @ < 16: get top of current ROMC
                    @ table; global variable which is
                    @ changeable per context
ldrlt         r1, [lr, r0, lsl #2]   @ < 16: read function pointer from
                    @ table assumed an array of pointers
                    @ patch functions
strlt          r1, [sp, #8]     @ < 16: store function pointer onto
                    @ stack in position of link register
ldmltfd       sp!, {r0-r1,pc}^      @ < 16: "fake" return from SWI, will
                    @ vector core to appropriate patch
                    @ function and set core back to previous
                    @ mode of operating
ldr          r1, =swi_hdler   @ >= 16: pointer to standard SWI
                    @ handler
mov        lr, pc         @ >= 16: set link register
bx         r1          @ >= 16: jump to standard SWI
                    @ handler
ldmfd       sp!, {r0-r1,pc}^      @ >= 16: pop registers from stack
```

52.4.2.5 External Boot Feature

Following a Reset event, the ARM issues an instruction fetch of the Reset Vector from address 0x0. This instruction, normally residing in ROM is usually a branch to a Reset handler or boot code which also normally resides in ROM. The ROMC external boot feature allows the bypassing of this code, using a different boot code residing perhaps in external memory.

This feature uses the data fix mechanism and works as follows: if the boot_int signal is negated when a Reset event occurred, the ROMC will perform a data fix of the Reset Vector at 0x0 with the following instruction (opcode 0xE59FF00C):

```
ldr         pc,[pc,#12]      @ read 0x0000_0014 for reset_vector
```

The value of PC when this instruction is executed is 8 so that a PC relative offset of 12 makes the source address 20 or 0x14. When this instruction executes, the ARM core reads from address 0x0000_0014, triggering a ROMC data fix operation which places the

value taken from the external boot address on the read data bus, with the two LSBs zeroed out. This value is returned to the ARM to be placed in the PC causing code fetch and execution to start from that address.

52.4.2.6 Alternate Masters and ROMC

The ROMC sits on the AHB bus of the internal ROM (ROMC). This means that the ROMC can modify values on the read data bus going to the master. Therefore, any master which reads an opcode patched or data patched location will read patched data.

52.5 ROMCP Memory Map/Register Definition

All registers are accessible through an IP Bus and can only be accessed in privileged mode. These registers can only be written with 32-bits stores and are clocked by hclk_reg.

The ROMC register placement was originated from the AWPT design used in the ARM7 platform of Neptune

ROMC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21A_C0D4	ROMC Data Registers (ROMC_ROMPATCH0D)	32	R/W	0000_0000h	52.5.1/4484
21A_C0D8	ROMC Data Registers (ROMC_ROMPATCH1D)	32	R/W	0000_0000h	52.5.1/4484
21A_C0DC	ROMC Data Registers (ROMC_ROMPATCH2D)	32	R/W	0000_0000h	52.5.1/4484
21A_C0E0	ROMC Data Registers (ROMC_ROMPATCH3D)	32	R/W	0000_0000h	52.5.1/4484
21A_C0E4	ROMC Data Registers (ROMC_ROMPATCH4D)	32	R/W	0000_0000h	52.5.1/4484
21A_C0E8	ROMC Data Registers (ROMC_ROMPATCH5D)	32	R/W	0000_0000h	52.5.1/4484
21A_C0EC	ROMC Data Registers (ROMC_ROMPATCH6D)	32	R/W	0000_0000h	52.5.1/4484
21A_C0F0	ROMC Data Registers (ROMC_ROMPATCH7D)	32	R/W	0000_0000h	52.5.1/4484
21A_C0F4	ROMC Control Register (ROMC_ROMPATCHCNTL)	32	R/W	0840_0000h	52.5.2/4485
21A_C0F8	ROMC Enable Register High (ROMC_ROMPATCHENH)	32	R	0000_0000h	52.5.3/4486
21A_C0FC	ROMC Enable Register Low (ROMC_ROMPATCHENL)	32	R/W	0000_0000h	52.5.4/4486
21A_C100	ROMC Address Registers (ROMC_ROMPATCH0A)	32	R/W	0000_0000h	52.5.5/4487
21A_C104	ROMC Address Registers (ROMC_ROMPATCH1A)	32	R/W	0000_0000h	52.5.5/4487
21A_C108	ROMC Address Registers (ROMC_ROMPATCH2A)	32	R/W	0000_0000h	52.5.5/4487
21A_C10C	ROMC Address Registers (ROMC_ROMPATCH3A)	32	R/W	0000_0000h	52.5.5/4487

Table continues on the next page...

ROMC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21A_C110	ROMC Address Registers (ROMC_ROMPATCH4A)	32	R/W	0000_0000h	52.5.5/4487
21A_C114	ROMC Address Registers (ROMC_ROMPATCH5A)	32	R/W	0000_0000h	52.5.5/4487
21A_C118	ROMC Address Registers (ROMC_ROMPATCH6A)	32	R/W	0000_0000h	52.5.5/4487
21A_C11C	ROMC Address Registers (ROMC_ROMPATCH7A)	32	R/W	0000_0000h	52.5.5/4487
21A_C120	ROMC Address Registers (ROMC_ROMPATCH8A)	32	R/W	0000_0000h	52.5.5/4487
21A_C124	ROMC Address Registers (ROMC_ROMPATCH9A)	32	R/W	0000_0000h	52.5.5/4487
21A_C128	ROMC Address Registers (ROMC_ROMPATCH10A)	32	R/W	0000_0000h	52.5.5/4487
21A_C12C	ROMC Address Registers (ROMC_ROMPATCH11A)	32	R/W	0000_0000h	52.5.5/4487
21A_C130	ROMC Address Registers (ROMC_ROMPATCH12A)	32	R/W	0000_0000h	52.5.5/4487
21A_C134	ROMC Address Registers (ROMC_ROMPATCH13A)	32	R/W	0000_0000h	52.5.5/4487
21A_C138	ROMC Address Registers (ROMC_ROMPATCH14A)	32	R/W	0000_0000h	52.5.5/4487
21A_C13C	ROMC Address Registers (ROMC_ROMPATCH15A)	32	R/W	0000_0000h	52.5.5/4487
21A_C208	ROMC Status Register (ROMC_ROMPATCHSR)	32	w1c	0000_0000h	52.5.6/4488

52.5.1 ROMC Data Registers (ROMC_ROMPATCH*n*D)

The ROMC data registers (ROMC_ROMPATCH7D through ROMC_ROMPATCH0D) store the data to use for the 8 1-word data fix events. Each register is associated with an address comparator (7 through 0). When a data fixing event occurs, the value in the data register corresponding to the comparator that has the address match is put on the romc_hrdata[31:0] bus until romc_hready is asserted by the ROM controller to terminate the access. A MUX external to the ROMC will select this data over that of romc_hrdata[31:0] in returning read data to the ARM core. The selection is done with the control bus rompatch_romc_hrdata_ovr[1:0] with both bits asserted by the ROMC.

If more than one address comparators match, the highest-numbered one takes precedence, and the value in corresponding data register is used for the patching event.

Address: 21A_C000h base + D4h offset + (4d × i), where i=0d to 7d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATAX																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ROMC_ROMPATCH*n*D field descriptions

Field	Description
DATAX	Data Fix Registers - Stores the data used for 1-word data fix operations. The values stored within these registers do not affect the writes to the memory system. They are selected over the read data from ROM when a data fix event occurs.

ROMC_ROMPATCH*n*D field descriptions (continued)

Field	Description
	If any part of the 1-word data fix is read, then the entire word is replaced. Therefore, a byte or half-word read will cause the ROMC to replace the entire word. The word is word address aligned.

52.5.2 ROMC Control Register (ROMC_ROMPATCHCNTL)

The ROMC control register (ROMC_ROMPATCHCNTL) contains the block disable bit and the data fix enable bits. The block disable bit provides a means to disable the ROMC data fix and opcode patching functions, even when the address comparators are enabled. The External Boot feature is not affected by this bit. The eight data fix enable bits (0 through 7), when set, assign the associated address comparators to data fix operations

NOTE

Bits 27 and 22 always read as 1s.

Address: 21A_C000h base + F4h offset = 21A_C0F4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved		DIS	Reserved												
Reset	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								DATAFIX							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ROMC_ROMPATCHCNTL field descriptions

Field	Description
31–30 -	This field is reserved. Reserved
29 DIS	ROMC Disable -- This bit, when set, disables all ROMC operations. This bit is used to enable secure operations. 0 Does not affect any ROMC functions (default) 1 Disable all ROMC functions: data fixing, and opcode patching
28–8 -	This field is reserved. Reserved
DATAFIX	**Data Fix Enable - Controls the use of the first 8 address comparators for 1-word data fix or for code patch routine.** 0 Address comparator triggers a opcode patch 1 Address comparator triggers a data fix

52.5.3 ROMC Enable Register High (ROMC_ROMPATCHENH)

The ROMC enable register high (ROMC_ROMPATCHENH) and ROMC enable register low (ROMC_ROMPATCHENL) control whether or not the associated address comparator can trigger a opcode patch or data fix event. This implementation of the ROMC only has 16 comparators, therefore ROMC_ROMPATCHENH and the upper half of ROMC_OMPATCHENL are read-only. ROMC_ROMPATCHENL[15:0] are associated with comparators 15 through 0. ROMC_ROMPATCHENLH[31:0] would have been associated with comparators 63 through 32.

Address: 21A_C000h base + F8h offset = 21A_C0F8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ROMC_ROMPATCHENH field descriptions

Field	Description
-	This field is reserved. Reserved

52.5.4 ROMC Enable Register Low (ROMC_ROMPATCHENL)

The ROMC enable register high (ROMC_ROMPATCHENH) and ROMC enable register low (ROMC_ROMPATCHENL) control whether or not the associated address comparator can trigger a opcode patch or data fix event. This implementation of the ROMC only has 16 comparators, therefore ROMC_ROMPATCHENH and the upper half of ROMC_ROMPATCHENL are read-only. ROMC_ROMPATCHENL[15:0] are associated with comparators 15 through 0. ROMC_ROMPATCHENLH[31:0] would have been associated with comparators 63 through 32.

Address: 21A_C000h base + FCh offset = 21A_C0FCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																ENABLE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ROMC_ROMPATCHENL field descriptions

Field	Description
31–16 Reserved	This field is reserved.
ENABLE	**Enable** Address **Comparator -** This bit enables the corresponding address comparator to trigger an event. 0 Address comparator disabled 1 Address comparator enabled, ROMC will trigger a opcode patch or data fix event upon matching of the associated address

52.5.5 ROMC Address Registers (ROMC_ROMPATCH*n*A)

The ROMC address registers (ROMC_ROMPATCHA0 through ROMC_ROMPATCHA15) store the memory addresses where opcode patching begins and data fixing occurs. The address registers ROMC_ROMPATCHA0 through ROMC_ROMPATCHA15 are each 21 bits wide and dedicated to one 4 Mbyte memory space. Bits 21 through 2 are address bits, to be compared with romc_haddr[21:2] for a match; bit 1 is also an address bit used for half word selection. Bit 0 is the mode bit (set to 1 for THUMB mode). 1-word data fixing can only be used on the first 8 of the address comparators. ROMC_ROMPATCHA0 through ROMC_ROMPATCHA15 are associated each with address comparators 0 through 15.

Address: 21A_C000h base + 100h offset + (4d × i), where i=0d to 15d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								ADDRX							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDRX															THUMBX
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ROMC_ROMPATCH*n*A field descriptions

Field	Description
31–23 -	This field is reserved. Reserved

Table continues on the next page...

ROMC_ROMPATCH*n*A field descriptions (continued)

Field	Description
22–1 ADDRX	Address Comparator Registers - Indicates the memory address to be watched. All 16 registers can be used for code patch address comparison. Only the first 8 registers can be used for a 1-word data fix address comparison. Bit 1 is ignored if data fix. Only used in code patch
0 THUMBX	THUMB Comparator Select - Indicates that this address will trigger a THUMB opcode patch or an ARM opcode patch. If this watchpoint is selected to be a data fix, then this bit is ignored as all data fixes are 1-word data fixes. 0 ARM patch 1 THUMB patch (ignore if data fix)

52.5.6 ROMC Status Register (ROMC_ROMPATCHSR)

The ROMC status register (ROMC_ROMPATCHSR) indicates the current state of the ROMC and the source number of the most recent address comparator event.

Address: 21A_C000h base + 208h offset = 21A_C208h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved														SW	Reserved
W															w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved										SOURCE					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ROMC_ROMPATCHSR field descriptions

Field	Description
31–18 -	This field is reserved. Reserved
17 SW	ROMC AHB Multiple Address Comparator matches Indicator - Indicates that multiple address comparator matches occurred. Writing a 1 to this bit will clear this it. 0 no event or comparator collisions 1 a collision has occurred
16–6 -	This field is reserved. Reserved
SOURCE	ROMC Source Number - Binary encoding of the number of the address comparator which has an address match in the most recent patch event on ROMC AHB. If multiple matches occurred, the highest priority source number is used.

Table continues on the next page...

ROMC_ROMPATCHSR field descriptions (continued)

Field	Description
	0 Address Comparator 0 matched 1 Address Comparator 1 matched 15 Address Comparator 15 matched

Chapter 53
Serial Advanced Technology Attachment Controller (SATA)

53.1 Introduction

The chip includes an integrated Serial Advanced Technology Attachment (SATA) Controller that is compatible with the Advanced Host Controller Interface (AHCI) specification.

The SATA Controller block (SATA) along with integrated physical link hardware (SATA PHY) provide one SATA port for the attachment of external SATA compliant storage devices.

The following section introduces the block features and architecture.

53.1.1 Features

The SATA block supports the following features:

- Compliant with the following specifications:
  - Serial ATA 3.0
  - AHCI Revision 1.3
  - AMBA 2.0 from ARM
- SATA 1.5 Gb/s and SATA 3.0 Gb/s speed.
- eSATA (external analog logic also needs to support eSATA)
- RX data buffer for recovered clock systems
- Data alignment circuitry when RX data buffer is also included
- OOB signaling detection and generation
- 8b/10b encoding/decoding
- Asynchronous signal recovery, including retry polling
- Power management features including automatic partial-to-slumber transition
- BIST loopback modes

- Supports one SATA device(port0)
- Configurable AMBA AHB interface (one master and one slave for each interface)
- Internal DMA engine
- Hardware-assisted Native Command Queuing for up to 32 entries
- Port Multiplier with command-based switching
- Disabling RX and TX Data clocks during power down modes

53.1.2 System Overview

SATA is an AHCI-compliant Host Bus Adapter (HBA). Together with the corresponding physical layer (PHY), it forms a complete AHCI HBA interface.

Note that only Port 0, PHY0, and bus interface are implemented in this product.

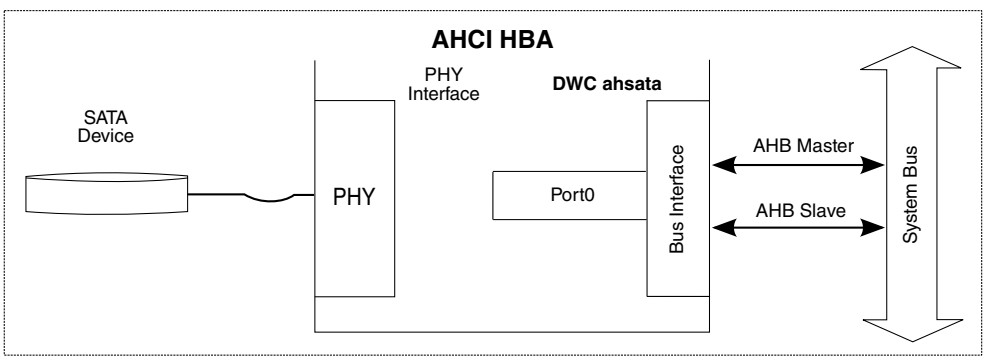


Figure 53-1. SATA AHCI System Block Diagram

53.2 Block Overview

This section describes the functional sub-blocks of the SATA block.

53.2.1 Block Diagram

The following figure provides a high-level view of the SATA architecture, followed by a brief description. The diagram shows additional ports and PHY interfaces not implemented in this product.

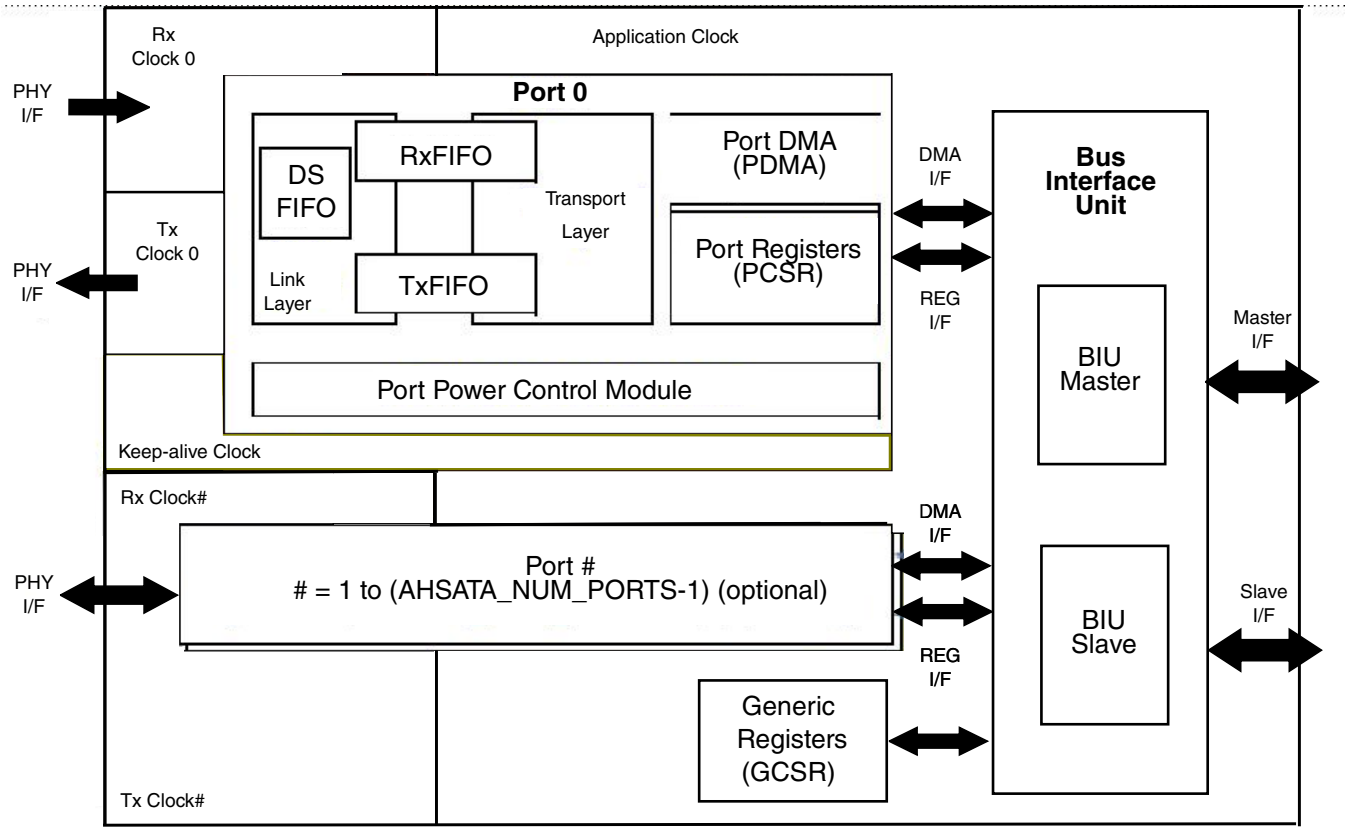


Figure 53-2. SATA Block Diagram

The SATA block consists of three main sub-blocks:

- Bus Interface Unit (BIU)
- Generic Registers (GCSR) (GCSR)
- Port

The system bus provides the application clock (hclk). BIU, GCSR and part of the Port operate in the application clock domain. Rx (when present), and Tx clocks are generated in the PHY and depend on the interface speed and PHY data width, as shown in the table below. The Rx OOB clock frequency is 50 or 60 MHz depending on PHY's reference clock (external or internal respectively).

Table 53-1. PHY-Supplied Clock Frequencies (clk rbc and clk asic)

Interface Speed (GB/s)	16/20 bits (Mhz)
1.5	75
3.0	150

Port logic contains its own DMA engine that implements AHCI functionality and initiates all of the data transfers between the external SATA device and system memory.

The Port PHY interface operates in three or four clock domains: RxOOB clock (clk_rxoob), Rx clock (clk_rbc) when present, keep-alive clock (clk_pmalive) and Tx clock (clk_asic). Most of the Link Layer (both receive and transmit data paths), and part of the Transport Layer always operate in the Tx clock domain. The Rx clock is a clock recovered from the PHY, and is used for clocking data into the SATA block. The Port contains a Power Control Module operating in the always-alive pmalive clock domain. This Power Control Module facilitates disabling Rx and Tx clocks in power down modes.

All SATA block clocks are asynchronous to each other in general. The BIU connects the Port to the system bus. AHB master interface is used to transfer data between the Port and the system memory, while the AHB slave interface is used by the software (driver) to access SATA block registers. All the SATA global registers are implemented in the Generic Register module.

53.2.2 SATA Block Transfer Hierarchy

The following figure shows the SATA block transfer hierarchy from the system bus point of view.

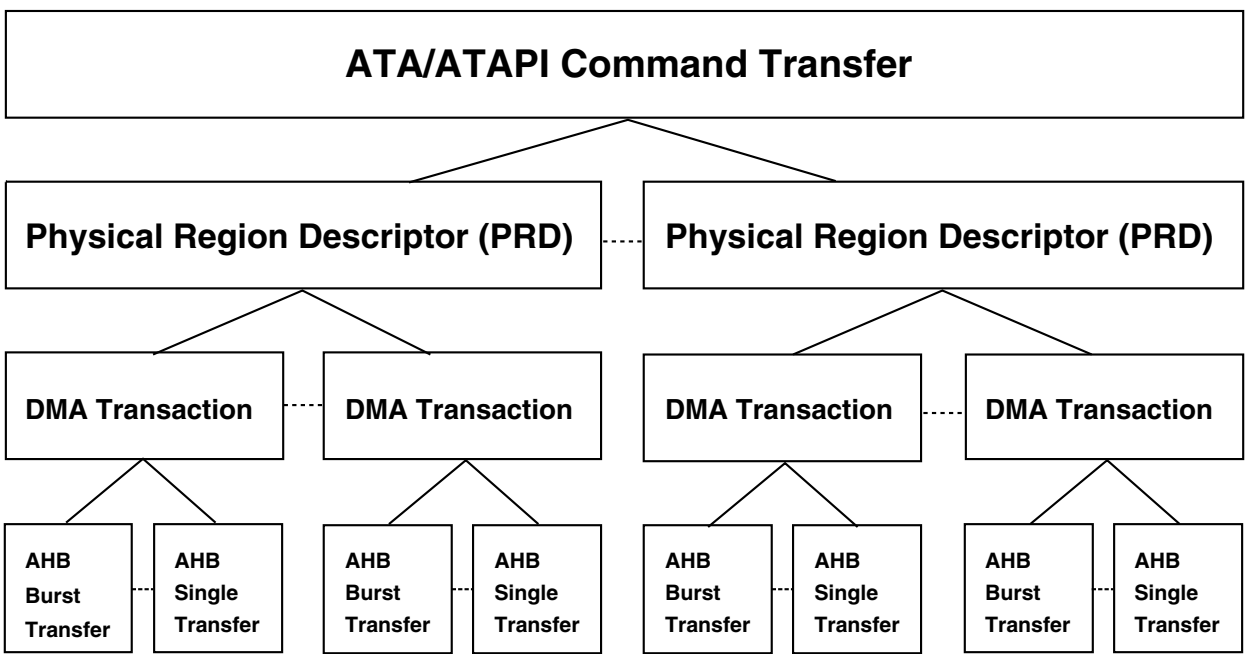


Figure 53-3. SATA Block Transfer Hierarchy

Data for an ATA/ATAPI command resides in the system memory in the form of one or more Physical Region Descriptors (PRD). PRD entries describe the physical location and length of data to be transferred.

Each PRD entry is read by the SATA Port DMA from the system memory before transferring the block of data associated with this entry.

The Port DMA transfers the PRD data block between the system memory and its FIFOs using one or more DMA transactions. DMA transaction size can be set by software.

Finally, the BIU Master generates one or more AHB burst or single bus transfers based on the DMA transaction request from the PDMA. The BIU Master tries to generate a burst equal to the DMA transaction size, but it might break it into several burst/single transfers due to various bus conditions (for example, early burst termination, 1-KB address boundary crossing, etc.).

53.2.3 Standards Compliance

1. The Serial ATA specifications can be found at the following website:

   http://sata-io.org

2. The AMBA Specification can be found at the following website:

   http://www.arm.com/products/solutions/AMBA_Spec.html

3. The AHCI specification can be found at the following website:

   http://www.intel.com/technology/serialata/ahci.htm

In the event of any conflict between the information in this chapter and the AHCI specification, the AHCI specification prevails.

53.3 Architecture

This section describes the SATA block interfaces, protocols, functionality, and implementation.

53.3.1 Architecture Overview

The SATA block diagram is shown in the following figure. While this diagram shows multiple ports, note that only one port is implemented.

The functional blocks that make up SATA are described in the following sub-sections.

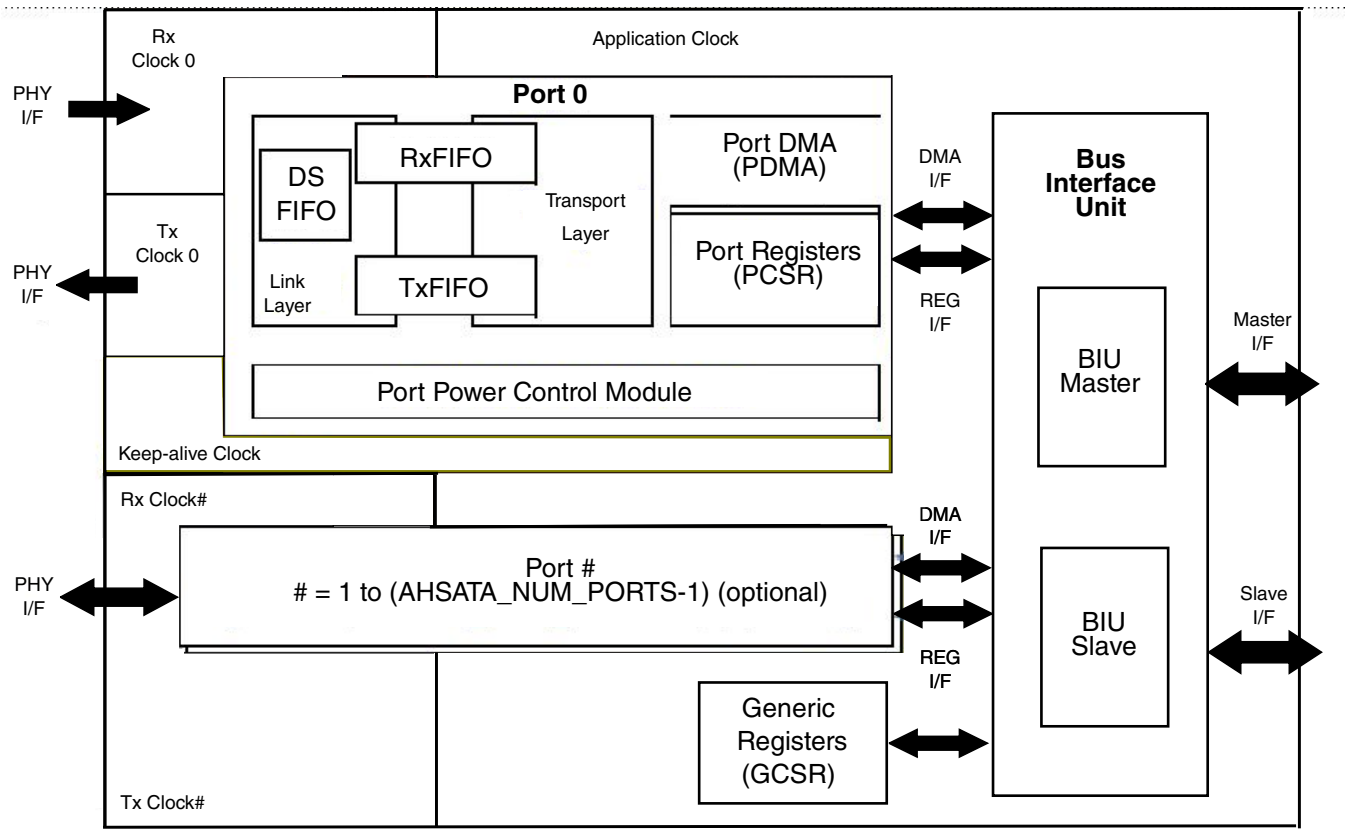


Figure 53-4. SATA Block Functional Diagram

53.3.2 Bus Interface Unit

The Bus Interface Unit (BIU) connects the AHB Master to the internal Port DMA controllers, and connects the AHB Slave to the internal Port registers.

The BIU is comprised of the following:

- AHB Slave Bus/GIF Interface (AHB Slave) and mapping logic.
- Register Read MUX, mapped to the AHB Slave read-response channel.
- AHB Master Bus/GIF Interface (AHB Master) and mapping logic.
- DMA Arbiter, mapped to the AHB Master.

The following figure displays a block diagram of the Bus Interface Unit.

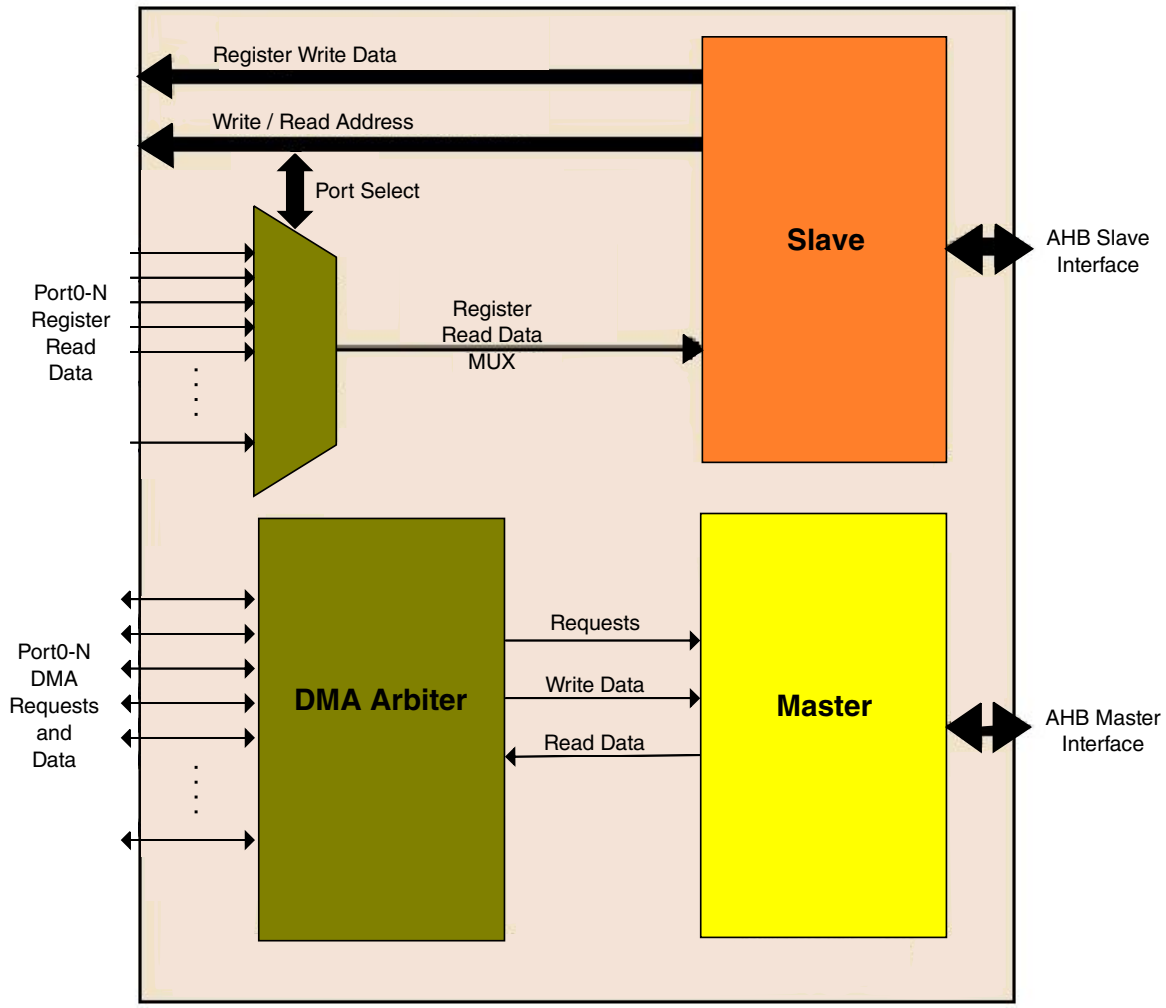


Figure 53-5. SATA Block Bus Interface Unit Block Diagram

53.3.2.1 AHB Slave Bus GIF Interface

The AHB Slave Bus GIF Interface (AHB Slave) converts AHB bus cycles to internal read/write request and response channel signals, which are mapped to Generic and Port control registers (GCSR and PCSR).

Characteristics of this unit include the following:

- Fully AMBA 2.0-Compliant AHB slave interface
- Supports single transfer type
- Supports INCR burst type
- Configured for little-endian byte ordering
- Supports 32-bit data bus width and 32-bit address width

- Master supports 32-bit (burst/single) and 16-bit (single only) transfer sizes, slave supports 8, 16, and 32-bit transfer sizes using SINGLE, INCR4, INCR8, and INCR16 burst types.
- Supports master BUSY and early burst termination
- Generates OKAY or ERROR response (SPLIT and RETRY are not supported).

The wait state for different accesses are:

- 0 wait state for any NSEQ or SEQ write access, 1 wait state for any NSEQread access.
- 1 wait state for any read after write decode to the same address.

AHB Slave generates ERROR response on s_hresp output when it detects any of the following illegal accesses:

- Transfer size is not 8, 16, or 32-bit.
- Address is not 32-bit or greater aligned.

Each Port decodes the csr_waddr address to select the required register when AHB Slave external mapping asserts corresponding p0_csr_sel signal during register write access. When a Port is not configured, then its register locations should be treated as reserved: read accesses return zeros, write accesses have no effect.

Figure 3-3 shows various AHB Slave accesses.

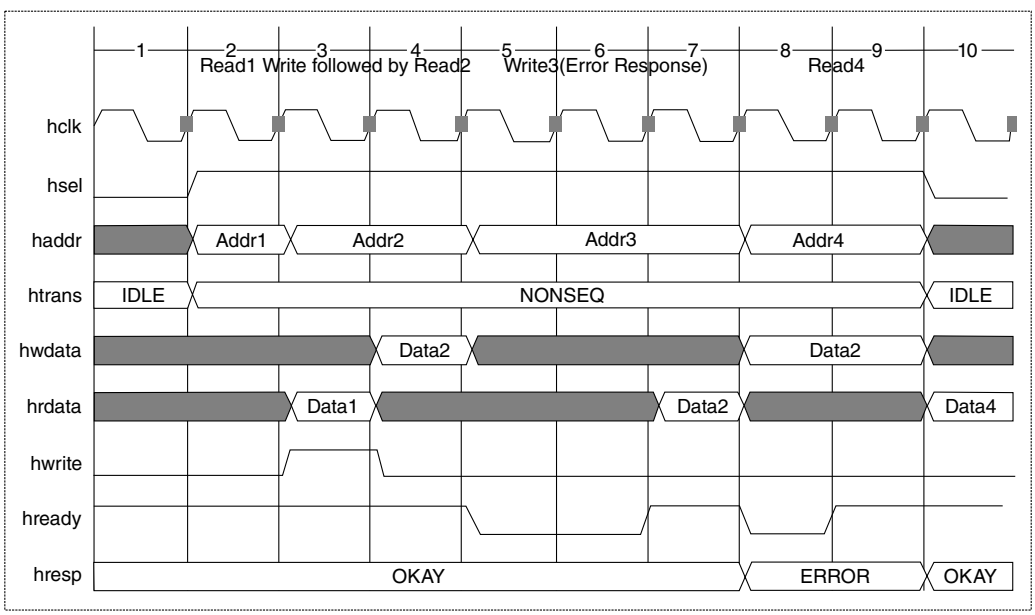


Figure 53-6. AHB Single Transfer

53.3.2.2 Register Read Multiplexer

The Register Read Multiplexer simply multiplexes all of the Port registers to the AHB Slave read response channel.

The AHB Slave read address selects which registers to read.

53.3.2.3 AHB Master Bus/GIF Interface

The AHB Master Bus/GIF Interface (AHB Master) converts Port DMA requests, from the DMA Arbiter, into AHB Master requests.

The AHB Master implements the following function:

- Converts Port DMA requests into AHB cycles

Characteristics of this unit include the following:

- Fully AMBA 2.0-compliant AHB master
- Burst types supported on the Port DMA request: SINGLE, INCR4, INCR8 & INCR16 burst
- Handles AHB SPLIT, RETRY, ERROR conditions
- Supports early burst termination via requesting remainder of burst
- Configured for little-endian byte ordering
- Handles AHB 1-KB boundary breaking
- All AHB transfers are 16- or 32-bit aligned
- Does not support locked transfers (m_hlock=0) and wrapped burst transfer (WRAP type).

The following sequence of events describes basic AHB Master operation:

1. When a Port becomes active, that is has data in the RxFIFO to transfer to system memory, or needs data to transfer to the Device, it asserts dma_req signal to the AHB Master to request DMA transaction in the direction indicated by the dma_read signal (0 - Port to AHB, 1- AHB to Port). Transaction size is indicated by the dma_count signal. Since all configured Ports operate independently, several Ports may assert dma_req at the same time.
2. The DMA Arbiter arbitrates between multiple active Ports using round-robin equal-priority scheme, selects a Port, and provides the request to the AHB Master. The AHB Master then latches starting address and transaction size. Arbitration is done on a per-transaction basis, that is, the Port "owns" the bus until all data of the given transaction size has been transferred or the transfer is terminated due to error.

3. AHB Master asserts either dma_rxpop signal to request data from the Port during AHB bus write transfer, or dma_txpush signal along with valid data to the Port during AHB bus read transfer.
4. AHB Master generates AHB transfer and tries to complete the data transfer either in one AHB burst (ideally), or it may break the transaction into multiple burst transfers, for example, due to the 1-KB address boundary crossing, or early burst termination condition.
5. AHB Master completes the AHB transfers when all the data of the requested transaction size has been successfully transferred, which allows the DMA Arbiter to switch to the next active Port. The following figure shows various BIU Master accesses.

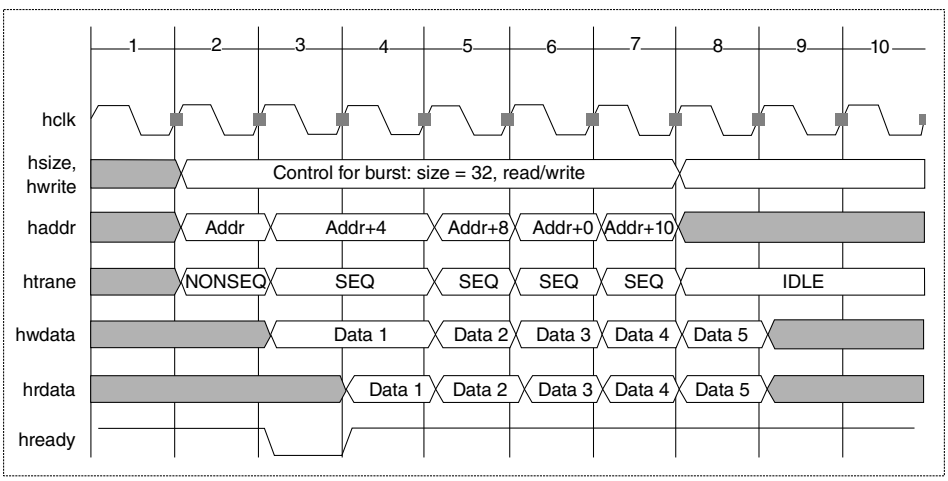


Figure 53-7. AHB Burst Transfer

53.3.2.4 DMA Arbiter

The DMA Arbiter simply monitors individual Port DMA requests and presents them to the AHB Master based on a round robin priority scheme.

Requests of larger burst length inherently gain more AHB bus priority than requests of shorter burst length. Software can use this approach to give Ports different priority.

53.3.3 Generic Registers (GCSR)

This module implements all SATA global registers and provides the following functions:

- Generic configuration and control;

- Global interrupt support;
- BIST operation (implementation-specific registers).

Refer to "Register Descriptions" for details on register operation.

53.3.4 Port

The Port instantiates the following modules:

- Port DMA
- Port Registers
- Transport Layer
- Link Layer
- Port Power Control Module

53.3.4.1 Port DMA

The Port DMA (PDMA) module implements the following functions:

- Monitors commands posted by system software using SATA_P0CI register. When any of the command slots becomes active, PDMA downloads the corresponding Register FIS from the Command List structure and passes it to the Transport Layer TxFIFO for transmission to the Device.
- Controls data transfer between the Transport Layer FIFOs and system memory using Physical Region Descriptor Tables (PRDT).
  - During Data FIS reception, PDMA requests AHB write transfer of SATA_P0DMACR[RXTS] size from the BIU Master when RxFIFO contains data of at least this size.
  - During Data FIS transmission, PDMA requests AHB read transfer of SATA_P0DMACR[TXTS] size from the BIU Master when TxFIFO contains space of at least this size.
- Transfers non-Data FISes received from the device to system memory using Received FIS Structure.

Most of the communication between the PDMA and software is done using two system memory descriptors that are constructed by software prior to initiating the transfer: FIS descriptor, which contains FISes received from the device, and the other is the Command List, which contains a list of 1 to 32 commands available for the Port to execute and the pointers for data transfers.

Some additional communication is done via registers located in the GCSR and PCSR modules.

System memory structures are described in the SATA AHCI specification and are not repeated in this document for the sake of brevity.

The PDMA module operates in the application clock (hclk) domain and has 32-bit-wide data path. It supports 32-bit and 64-bit (when M_HADDR_WIDTH=64) addressing.

The following figure shows a high-level state diagram of the Port DMA operation. Refer to the SATA AHCI specification for a more detailed Port state diagram.

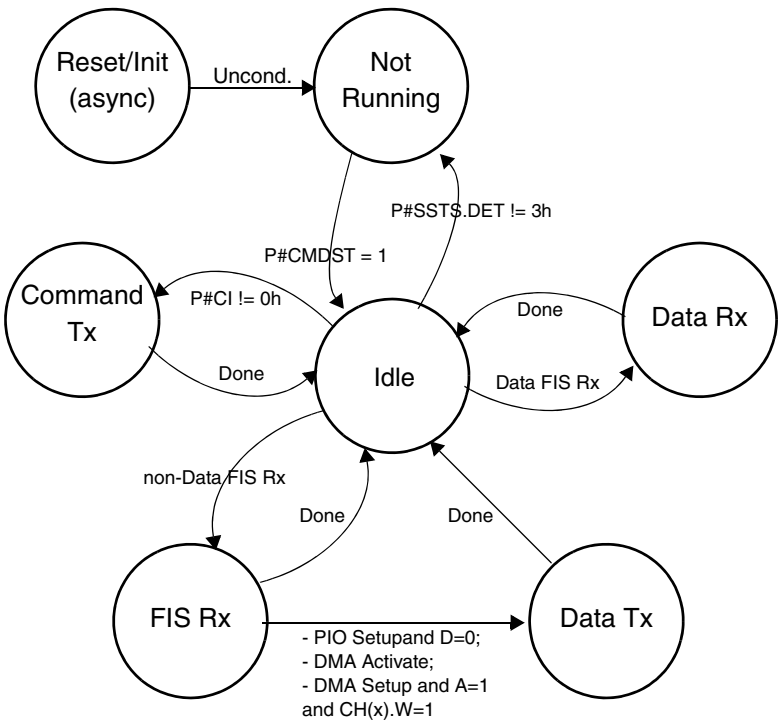


Figure 53-8. Port DMA State Diagram

53.3.4.2 Port Registers

The Port registers (PCSR) module implements all Port-specific registers:

- Command List and FIS Base address
- Interrupt Status/ Enable
- Port Command/ Status
- Task File Data/ Signature/ Serial ATA
- DMA status/control (implementation-specific)
- PHY status/control (implementation-specific)

Refer to "Register Descriptions" for details on register operation.

53.3.4.3 Transport Layer

The Transport Layer constructs Frame Information Structures (FIS) for transmission and decomposes received FISes.

The following list describes how an FIS is constructed:

- Receives FIS content from the Port DMA or Port Register modules;
- Notifies the Link Layer of required frame transmission and passes FIS to Link;
- Manages TxFIFO flow, notifies Link of required flow control via TxFIFO flags;
- Receives frame receipt acknowledge from Link;
- Reports good transmission or errors to requesting higher layer.

The following list describes how an FIS is decomposed:

- Receives the FIS from the Link Layer;
- Determines FIS type and checks for PHY/Link/Transport errors;
- Provides good/bad FIS acknowledge to the Link Layer;
- Distributes the FIS content to the locations indicated by the FIS type;
- Reports good reception or errors to the Port DMA/Registers.

The Transport Layer functional block diagram is shown in the figure below. The Transport Layer consists of the following five main modules:

- Receive FIFO (RxFIFO)
- Transmit FIFO (TxFIFO)
- Transport Check module (TCHK)
- Transport State Machine module (TSM)
- Synchronization module (APP_ASIC)

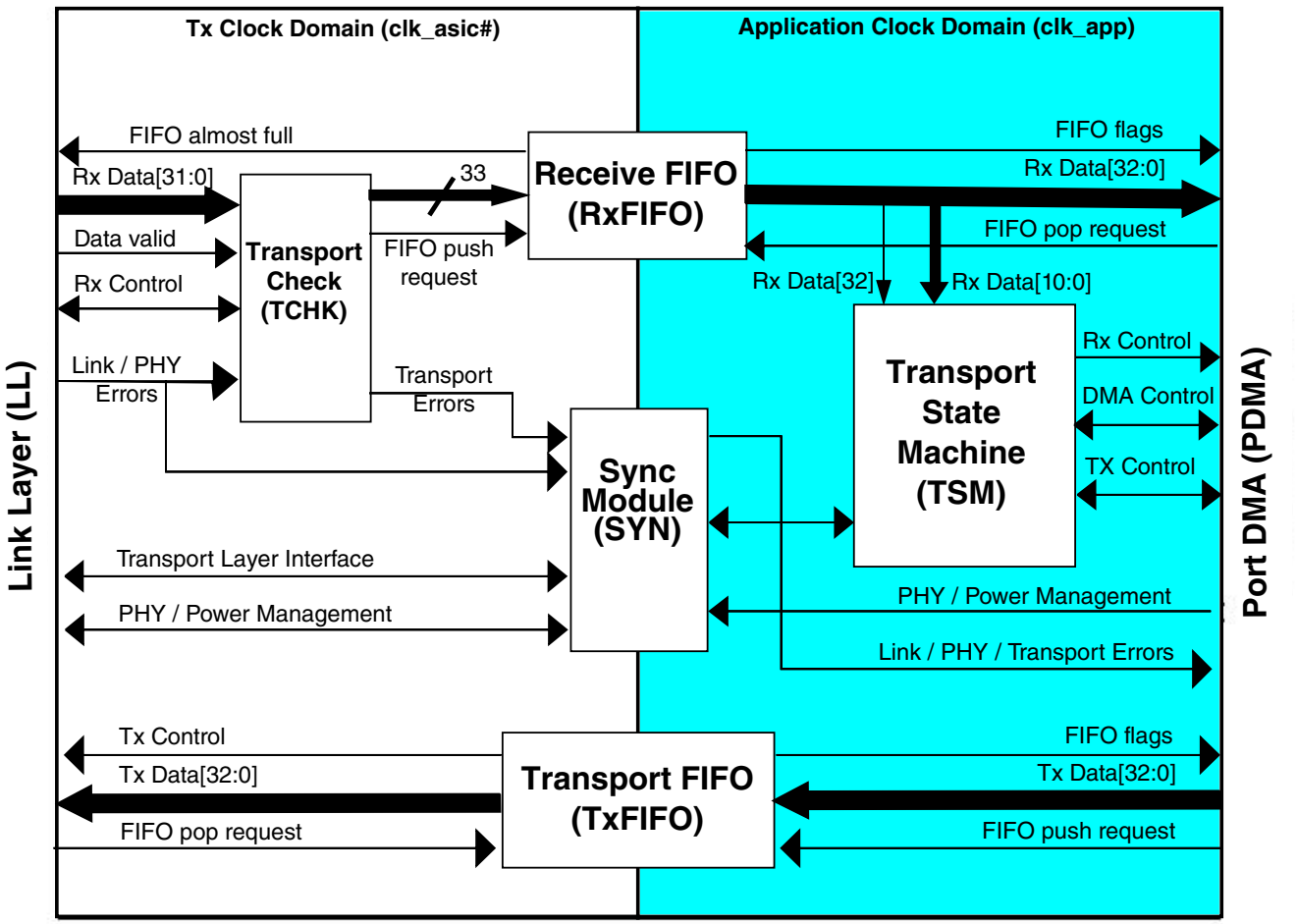


Figure 53-9. Transport Layer Functional Block Diagram

The Transport Layer operates in two clock domains: transmit and application. Transmit clock is generated in the PHY and depends on the Link Layer data path width (valid frequency values are: 37.5 MHz, 75 MHz,150 MHz, and 300 MHz). The application clock is sourced from the system bus and depends on the software. Both transmit and receive data paths are 32-bits wide.

The Transport Layer block provides FIS reception and transmission functions of the SATA Transport Layer. During reception the Transport Layer receives a new FIS from the Link Layer through the RxFIFO, decodes the FIS type, and instructs the PDMA to route the FIS payload data to the appropriate location in system memory. During transmission the Transport Layer instructs the PDMA to construct the appropriate FIS, and then passes it to the Link Layer through the TxFIFO. The Transport Layer block receives all the PHY/Link errors from the Link Layer, detects Transport errors, and passes them to the PCSR for setting the corresponding error bits.

The Transport Layer processes one FIS at time on the transmit side, meaning only one FIS is allowed in the TxFIFO at a time. On the receive side, RxFIFO can potentially contain more than one FIS at a time. For example, when the device transmits several DMA Data FISs back-to-back with minimal delay, RxFIFO might still have the previous Data FIS while the next FIS is being received.

53.3.4.3.1 Transport Layer FIS Reception

The FIS reception process is described as follows:

- The Link Layer starts frame reception and passes FIS content to the Transport Layer THCK. RxFIFO "almost full" flag notifies the Link Layer to send HOLDp to the device to prevent RxFIFO overflow. Upon detecting EOFp, the Link Layer asserts an "End status" signal to indicate the end of the FIS. All Link Layer/PHY errors are valid at this time.
- THCK module checks for Transport Layer protocol errors, passes FIS data to the RxFIFO, then appends "End Status" DWORD at the end with all the Link/PHY and Transport errors.
- TSM module receives the FIS from the RxFIFO and passes it to the PDMA/PCSR. When any of the Link/PHY/Transport errors is detected, then the FIS is either ignored (when non-Data FIS) or the transfer is aborted (when Data FIS) and the corresponding bits are set in the SATA_P 0SERR register.

53.3.4.3.2 Transport Layer FIS Transmission

The FIS transmission process is described as follows:

- The PDMA detects a request from the system software and notifies the TSM to enter a transmit state. The DMA data transmission is activated by the TSM after it receives DMA Activate FIS from the device.
- The PDMA receives the appropriate FIS from BIU Master and pushes it into the TxFIFO. The following FIS types are supported:
  - Register FIS - Control or Command type.
  - Data FIS - PIO or DMA type.
  - BIST Activate FIS
- The Link Layer uses negation of the TxFIFO "empty" flag to generate SOFp and begin frame transmission. Bit 32 of the TxFIFO is used to indicate the FIS "last DWORD" to the Link Layer. When the Link Layer sees this bit valid, it closes the frame with CRC and EOFp.
- The TSM waits for either positive or negative frame transmission acknowledgement from the Link Layer (Link Layer "handshake" error). Both of these conditions are passed from Link to TSM in the "End Status" DWORD. Negative acknowledgement

is generated when the device detects an error during the frame reception and signals it to the host Link Layer. In this case any non-data FIS is resent to the device using Transport Layer retry logic. When the error is detected during Data FIS transmission, then this transfer is aborted and the FIS is not resent.

NOTE

When neither positive nor negative acknowledgement is received from the Link Layer following frame transmission, host s/w times-out and resets the interface.

53.3.4.3.3 Error Handling

All SATA errors are summarized in SATA_POSERR. The Link Layer accumulates all Link Layer/PHY errors during frame reception and presents them to the Transport Layer TCHK module with "End Status" signal asserted when it detects EOFp. "PHY Not READY" and "Link Illegal Transition/Sequence" errors terminate the current frame reception/transmission in progress and cause PHY/Link Layer reset and "End Status" assertion. During frame transmission, Link Layer detects R_ERRp/R_OKp from the device and passes this condition in the "End Status" DWORD.

The TCHK module checks for Transport Layer errors in the received FIS when no PHY/Link Layer errors were detected. All errors from the PHY, Link Layer, and Transport Layer TCHK module are passed to the Transport Layer TSM module by way of RxFIFO in the "End status" DWORD (with bit 32 set).

Some errors such as "Non-recovered/Recovered Data Integrity" are detected in the Transport Layer TSM module. "Internal Host Adapter" error is detected in the BIU.

All PHY, link, and transport errors are also passed to the PCSR module through the APP_ASIC module directly for setting the appropriate bits in the SATA_P0SERR register. This is done to insure that error bits are reliably set in various error conditions. For example, when Data FIS is corrupted, this may result in potential DMA lock-up unless error that caused this condition is passed to the PCSR module and software can act on it.

PHY internal errors are filtered out outside the FIS. Errors related only to the current receive FIS cause SATA_P0SERR[DIAG_I] bit to be set. To enable errors inside the FIS or outside the FIS to be reflected in the SATA_P0SERR register, software must set SATA_BISTCR[ERREN] bit for the corresponding Port selected by the SATA_TESTR[PSEL] field.

53.3.4.3.4 Receive/Transmit FIFO (Rx/TxFIFO)

Both receive and transmit FIFOs are used as temporary FIS buffers and for clock domain crossing.

The RxFIFO width is 33 bits: 32 bits are used to transfer data and the 33rd bit is used to indicate the "End- Status" DWORD so the Transport Layer can detect the end of the previous FIS and the start of the next FIS in the situation when more than one FIS is in the RxFIFO.

The TxFIFO is 33-bits wide: 32 bits are used to transfer data and 33rd bit is used to indicate the "last" FIS DWORD to the Link Layer. Both FIFOs are reset on power-up either by the system bus reset signal, by software setting SControl register (P0SCTL) DET field bit 0 (which results in the SATA block performing an interface initialization sequence COMRESET), or by the COMINIT condition.

NOTE

Both RX and TX FIFO RAM width is (M_HDATA_WIDTH + M_HDATA_WIDTH/32 + 1) bits.

You can select FIFO capacity in DWORDS for each Port separately, based on the system bus software requirements, such as number of masters, burst size, utilization, and so on.

The RX FIFO “almost full” level is set to the value indicated in the following table, to prevent an RX FIFO overflow regardless of the total Host and Device HOLD-HOLDA latency. When the RX FIFO “almost full” flag is asserted, the Link layer starts sending a HOLD status to the Device.

NOTE

Both RX and TX FIFO RAM width is (M_HDATA_WIDTH + M_HDATA_WIDTH/32 + 1) bits.

Table 53-2. RX FIFO “Almost Full” Level

RX FIFO Capacity (DWORDS)	M_HDATA_WIDTH (Bits)	“Almost Full” Level (DWORDs)
64	32 64 128	54 52 52
128	32 64 128 256	62 60 56 64
256	32 64 128 256	64
512	32 64 128 256	64
1024	32 64 128 256	64
2048	32 64 128 256	64

NOTE

It is very important that the total HOLDp/HOLDAp latency for both host and device does not exceed the number of RX FIFO

almost full-level DWORDs as indicated in Table 53-2, otherwise, RX FIFO overflow (fatal error) occurs.

NOTE

Host latency must be calculated by adding the Link layer latency to the Host PHY latency. Some SATA configurations are close to 20 DWORDs leaving no margin for the PHY. It is recommended that such configurations be used only for FPGA validation/testing and not for actual product because exceeding 20 DWORD latency requirement may result in Device RX FIFO overflow.

53.3.4.4 Transport Check (TCHK)

The TCHK module provides the following functions:

- Detects new FIS reception by the Link Layer based on the received control signals.
- Decodes the FIS type located in the least-significant byte of the first DWORD and checks its validity. The following FIS types are supported:
  - Register FIS
  - Set Device Bits FIS
  - PIO Setup FIS
  - DMA Activate FIS
  - DMA Setup FIS
  - Data FIS
  - BIST Activate FIS
  - Unknown FIS (length is less than or equal to 64 bytes)
- Checks for all the Transport Layer errors (for example, unrecognized FIS, protocol, or transition).
- Detects an "End Status" signal assertion indicating the end of the current FIS from the Link Layer and passes all Link Layer/PHY/Transport Layer errors to the RxFIFO and to the PCSR module.
- The TCHK provides "Good FIS/Bad FIS" status acknowledgement to the Link Layer at the end of the received FIS.

The TCHK module receives 32-bit FIS DWORD data from the Link Layer and adds one bit (bit 32) before writing it to the RxFIFO. This bit indicates either FIS data, when cleared, or "End Status" DWORD, when set. The following Transport Layer errors are checked in the TCHK (assuming no errors were detected in the Link/PHY):

1. FIS length:
   - Non-data FIS according to the FIS type

    - Data FIS should be between 2 and 2049 DWORDs
    - Unknown FIS should be between 1 and 16 DWORDs
2. PIO Setup FIS transfer count - should be non-zero and even byte count and not exceed 8192 bytes
3. PIO Data FIS following the PIO Setup FIS with D=1 (PIO read) DWORD count - should match the transfer count
4. PIO read protocol FIS sequence - only Data FIS or end status when error are expected after the PIO Setup FIS with D=1, any other FIS would be negatively acknowledged to the Link Layer
5. DMA Setup FIS buffer offset - bits 0 and 1 should be cleared and transfer count should be an even (not zero) number
6. First Party DMA read protocol - DMA Setup FIS with D=1 is followed either by Data FIS or Set Device Bits FIS or end status when error
7. First Party DMA write protocol - DMA Setup FIS with D=0 is followed by DMA Activate FIS (when A=0) or Set Device Bits FIS or end status (when A=1)
8. BIST Activate FIS is supported type only (see BIST Operation for details)
9. RxFIFO push error for Data FIS - detected when Link has valid data and RxFIFO is "full" (for example, device violates HOLD latency requirement)

The Transport Transition Error SATA_P 0SERR[DIAG_T] bit is set when errors 1-8 are detected. The Unknown FIS SATA_P 0SERR[DIAG_F] bit is set when the Unknown FIS length does not exceed 64 bytes. The Protocol Error SATA_P 0SERR[ERR_P] bit is set on detection of error 9.

53.3.4.4.1 Transport State Machine (TSM)

The TSM module provides the following functions:

- Implements the host Transport Layer state machine according to the SATA spec with the exception of the FIS checking and error handling functions.
- Decodes the FIS type by reading the least-significant-byte of the first DWORD of the FIS.
- Detects the "End status" DWORD and checks for any Link Layer/PHY/Transport Layer errors. When any of the errors is detected:
    - On a non-data FIS, the received FIS is discarded, the transmitted FIS is retried indefinitely, and the corresponding SATA_P0SERR register ERR_I bit is set.
    - On a data FIS, it can be passed to the system memory before the final status is reflected in the SATA_P0SERR register ERR_T bit.
- Generates/receives the appropriate control signals to/from the PDMA based on the received FIS and its state.
- Handles transfer termination requests originated from the Link Layer or PDMA module.

53.3.4.4.2 Sync Module (APP_ASIC)

This module is used to synchronize several control signals between the Link Layer and the Transport Layer clock domains.

53.3.4.5 Link Layer

The Link Layer performs the following functions:

- Controls initialization between the Link Layer, PHY, and a connected device
- Generates and detects OOB signaling
- Transmits and receives frames
- Transmits primitives based on control signals from the Transport Layer and PHY
- Receives primitives from the PHY layer that are used to control the Transport and Link Layers
- Controls power management via the Port Power Control Module

Hot-plugging a device is digitally supported in the Link Layer when Tx OOB sequences are generated by the Link Layer via normal initialization polling. When Tx OOB sequences are generated in the PHY, it is up to the PHY to digitally support hot-plugging a device.

The Link Layer functional block diagram is shown in the following figure.

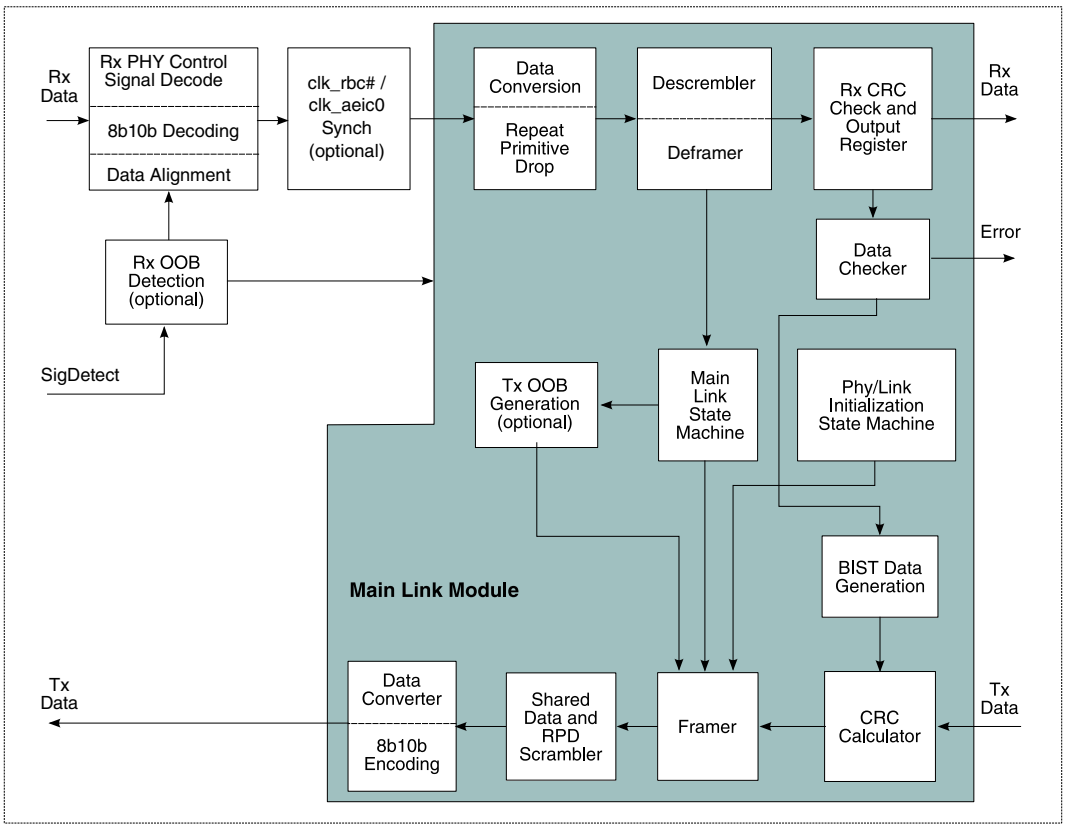


Figure 53-10. Link Layer Functional Block Diagram

On power-up, system reset or device hot-plug, the following sequence occurs:

1. The Link Layer transmits sequences of control data and ALIGN Primitives to the PHY .
2. They are then forwarded to a device PHY as OOB signaling.
3. In addition, the Link Layer detects OOB sequences.

These OOB sequences bring the host controller, PHY, and device to an initialized condition. Once this occurs:

1. The Link Layer passes a PHY Ready status to the Transport Layer and normal communication begins.
2. The Link Layer receives requests from the Transport Layer to transmit data, in the form of a Frame Information Structure (FIS) comprised of DWORDs, to a device via the local PHY.
3. The Link Layer in turn transmits the FIS by inserting Primitives, scrambling and optionally encoding the data, sending it to the PHY and waiting for status.

4. When a status FIS is received, the Link Layer optionally decodes, aligns and descrambles the data, removes Primitives and forwards the data to the Transport Layer.
5. The Link Layer then notifies the Transport Layer of the ending transfer status. The Link Layer has no notion of the FIS content, other than its beginning and end points and CRC.
6. Data alignment is performed on received FIS data via ALIGN Primitives. Flow control is also achieved on FIS going in either direction via HOLD Primitives.
7. In addition, the Link Layer receives requests from the Transport and PHY Layers to go into and out of power management modes.

Power management is achieved by notifying the PHY of a partial or slumber condition and then disabling normal data transmission on PHY Rx and Tx interfaces until a wake-up request from Transport Layer or remote device via the PHY is seen from the Power Control Module. Power management is controlled via Partial and Slumber requests as described in the SATA specifications.

The Initialization State Machine controls the Link Layer, PHY and device system initialization. The main Link Layer State Machine controls FIS traffic, flow control, and error detection and status reporting. FIS traffic is generated and disassembled via Framer and Deframer modules. The Link Layer also performs CRC calculations on FIS, as well as scrambling and optionally encoding the data.

- Decoding of received FIS is performed in the clk_rbc0 clock domain due to the fact that the incoming FIS is on an asynchronous, but frequency locked clock of the same rate as the clk_asic0 clock domain.
- 8b/10b encoding and decoding are performed in the Link Layer.

The Link Layer receives data on either clk_rbc0, recovered from the incoming data stream by the PHY, or on clk_asic0. This single receive clock is then used in this module to decode data and control signals from the PHY and pass it to the rest of the Link Layer. Data is passed through a synchronizing Datastream FIFO. ALIGN Primitives are also detected and dropped in the front end of the receiver as a means of guaranteeing no Datastream FIFO overruns, when a Datastream FIFO is included. ALIGN Primitives are also used to synchronize to the data stream in the PHY by triggering data realignment where necessary.

Finally, ALIGNs are required by the Tx OOB initialization state machine to complete initialization, following the SATA specifications. For this reason, the PHY must indicate the presence of at least two ALIGNs after the Link Layer detects the release of COMWAKE. Otherwise the Link Layer is not able to complete initialization and begin normal operation. This is required regardless whether the PHY drops ALIGNs at any other time.

NOTE

Even if the PHY drops ALIGNs, data indicating the comma character must be present on phy_rx_data, in the corresponding phy_comma_det slot. This is required to invalidate comma characters before they are stable.

This Databook attempts to avoid redundancy with the *High Speed Serialized ATA* specification. Please refer to the latter for standard specifications and descriptions. This discussion pertains to specific implementation-related operation and details.

53.3.4.5.1 Link Layer Features

The SATA block Link Layer features are as follows:

- Rx Data Buffer for recovered clock systems
- OOB signaling and system Initialization
- Frame negotiation and arbitration
- Envelope framing/deframing
- CRC calculating, insertion and checking
- 8b/10b encoding/decoding
- Flow control
- Frame acknowledgement and status reporting
- Data width conversions
- Data scrambling/descrambling for EMI reduction
- Repeat Primitive data transmission and reception handling
- ALIGN Primitive detection, dropping and data alignment
- Power management support

53.3.4.5.2 User-Defined Status and Control

There are up to 32 bits each of optional user-defined status and control Ports available in the SATA block. These are used to obtain information from the PHY, and control PHY functions that might differ across particular PHYs and which are not defined in the SATA specifications.

Each of these variable bit width, user defined Ports map to a register that can be read and/or written by the system via the Bus Interface. When no user-defined Ports are included in the design, they are absent from the netlist, along with the registers that would have been allocated for their use. See Figure 53-8 for the top-level I/O diagram.

NOTE

There are clock crossing restrictions on these Ports. Refer to "Signal Descriptions".

53.3.4.5.3 PHY Initialization Details

Please refer to the "Out of band signaling" section of the SATA specifications for the following descriptions.

The PHY/device signaling from SATA block is dependent on whether the Link Layer generates and detects the proper OOB sequences or whether the PHY generates and detects them.

NOTE

For PHY Initialization and general operation related to the Link Layer, all SATA block inputs and outputs are sampled and generated, respectively, in the following clock domains. Note that the arrangement of signals facilitates there being no clk_rbc present until the clock is recovered from incoming data.

- Rx Data Inputs
  - clk_asic
  - clk_rbc
    - phy_rx_err
    - phy_comma_det
    - phy_rx_data_vld
    - phy_rx_data
    - phy_rx_kch_det
    - phy_rx_dec_err
    - phy_rx_disp_err
- Tx Data Outputs
  - clk_asic
  - phy_tx_data_vld
  - phy_tx_data
- Control Outputs
  - clk_asic
    - phy_rx_enable
    - phy_tx_enable
    - phy_spdsel
    - phy_nearafelb
    - phy_farafelb
    - phy_reset
- Control Outputs
  - clk_pmalive
    - phy_partial
    - phy_slumber

- Control Inputs - Fixed, multiple and/or configuration-dependent clock domains
  - phy_sig_det - Sampled in all of these clock domains
    - clk_rbc
    - clk_rxoob
    - clk_pmalive
    - clk_asic
- phy_calibrated Sampled in all of these clock domains
  - clk_rbc
  - clk_asic
- phy_spdmode
  - clk_asic

NOTE

phy_rx_data_vld should not be delayed after OOB initialization (held inactive LOW), while phy_comma_det is indicating COMMA characters are being received, or normal operation may not begin. Either phy_comma_det must be ANDed with phy_rx_data_vld, or all phy_rx_data_vld bits must be tied permanently HIGH.

However, in order to tie phy_rx_data_vld permanently HIGH, a phy_sig_det must be present and indicate valid signal detection. For example, phy_sig_det cannot be tied HIGH when phy_rx_data_vld is also tied HIGH. Normal operation can fail due to missing the first non-ALIGN Primitives when they are in the form of Repeat Primitive Data, before phy_rx_data_vld becomes active. A PHY that always drops all ALIGN data cannot delay phy_rx_data_vld at all after initialization. Finally, phy_rx_data_vld and phy_comm_det should not become high until the PHY has completely locked onto ALIGNs and data is error free, to avoid problems transitioning between OOB and normal operation.

53.3.4.5.3.1 Link Layer Tx OOB Initialization Sequence Details

NOTE

In order for the Link Layer to generate the Tx OOB initialization sequences, at least two ALIGN Primitives must be indicated and flagged to the Link Layer by the PHY via the phy_comma_det signal. This must occur after the release of COMWAKE, per the SATA initialization specifications. Otherwise the Link layer is not able to complete initialization

and begin normal operation. This is required regardless of whether the PHY drops ALIGNs at any other time. In addition, if data can ever become misaligned from the PHY, ALIGN Primitives are required in order for the Link Layer to realign the data. Finally, ALIGNs must be returned to the SATA block for all BIST modes to function properly.

The Link Layer Tx OOB initialization sequence is depicted in the figure below. Not all details are shown, but the following sequence summarizes the flow:

1. PHY Reset state is entered during a system asynchronous-reset or if a Port reset (COMRESET) has been detected. This causes the phy_reset output signal to become active and can be used to synchronously clear the PHY of any errors, when the PHY has that capability. The PHY reset is active for 12 1.5 Gb/s rate DWORDs of time, or 320 ns.

NOTE

clk_asic0 cannot be removed while phy_reset is asserted, because clk_asic0 is required to negate phy_reset.

Signal phy_reset is intended only to clear non SATA errors in this configuration, and must not be used to asynchronously reset the PHY.

2. Upon exiting PHY Reset state, the PHY speed is always set to 1.5 Gb/s speed, regardless of the speed being negotiated for.
3. After a PHY reset, and anytime an unsolicited COMINIT has been detected, the PHY Wait state is entered in order to check and wait for PHY calibration.
4. After the PHY Wait state, communication with a device is attempted by transmitting a COMRESET sequence, followed by waiting for a COMINIT sequence from the device. This takes place in a polling routine. During the Wait Cominit state, when a COMINIT has not been detected in 14ms, the state machine loops back and restarts from the PHY Wait state indefinitely.
5. Once a COMINIT has been detected, normal OOB sequences then take place, interrupted only by an unsolicited COMINIT; a COMINIT that was not expected.
6. After completion of the OOB sequence, when 6.0 Gb/s or 3.0 Gb/s are enabled and have not already been attempted and failed, phy_spd_sel is asserted to the PHY to change clk_asic0 rate to the highest supported speed. The host then begins waiting for ALIGNs.
7. At the beginning of awaiting ALIGNs, the speed is changed back to the highest speed to be negotiated. Otherwise the speed stays at Gen1. During the Await ALIGN state, when an ALIGN has not been detected by the time 32K 1.5 Gb/s rate DWORDS of D10.2 characters have been transmitted, the Link Layer moves to the

PHY Wait state. In addition, when a higher speed fails negotiation, an internal flag is set to force the host to stay at the current speed after the next OOB sequence has completed, unless one of the following occurs:

- Asynchronous system reset
- Port reset
- An unsolicited COMINIT has been detected from the device
- 6.0 and 3.0 speed disabled and re-enabled (programmed to 1.5 Gb/s speed, then back to 3.0 Gb/s or 6.0 Gb/s, followed by a host COMRESET)

8. When an ALIGN has been detected before 32K 1.5 Gb/s rate DWORDS of D10.2 have been transmitted, the Link Layer moves to the Send ALIGN state, followed by the Init Complete state, once three non-ALIGN Primitives have been detected. Note that while the Link Layer is generating D10.2 characters at 3.0 Gb/s or 6.0 Gb/s, 1.5 Gb/s rate D10.2 characters are emulated by doubling or quadrupling the transmitted data, i.e., 1100110011, as opposed to 1010101010.
9. Once initialization is complete, the Ready state is entered and the Main Link state machine takes control.
10. Finally, the Partial and Slumber states are only entered after a Power Mode has been set. This disables the Tx interface until either the host or device requests a wake-up. For further information, refer to Power Management Operations.

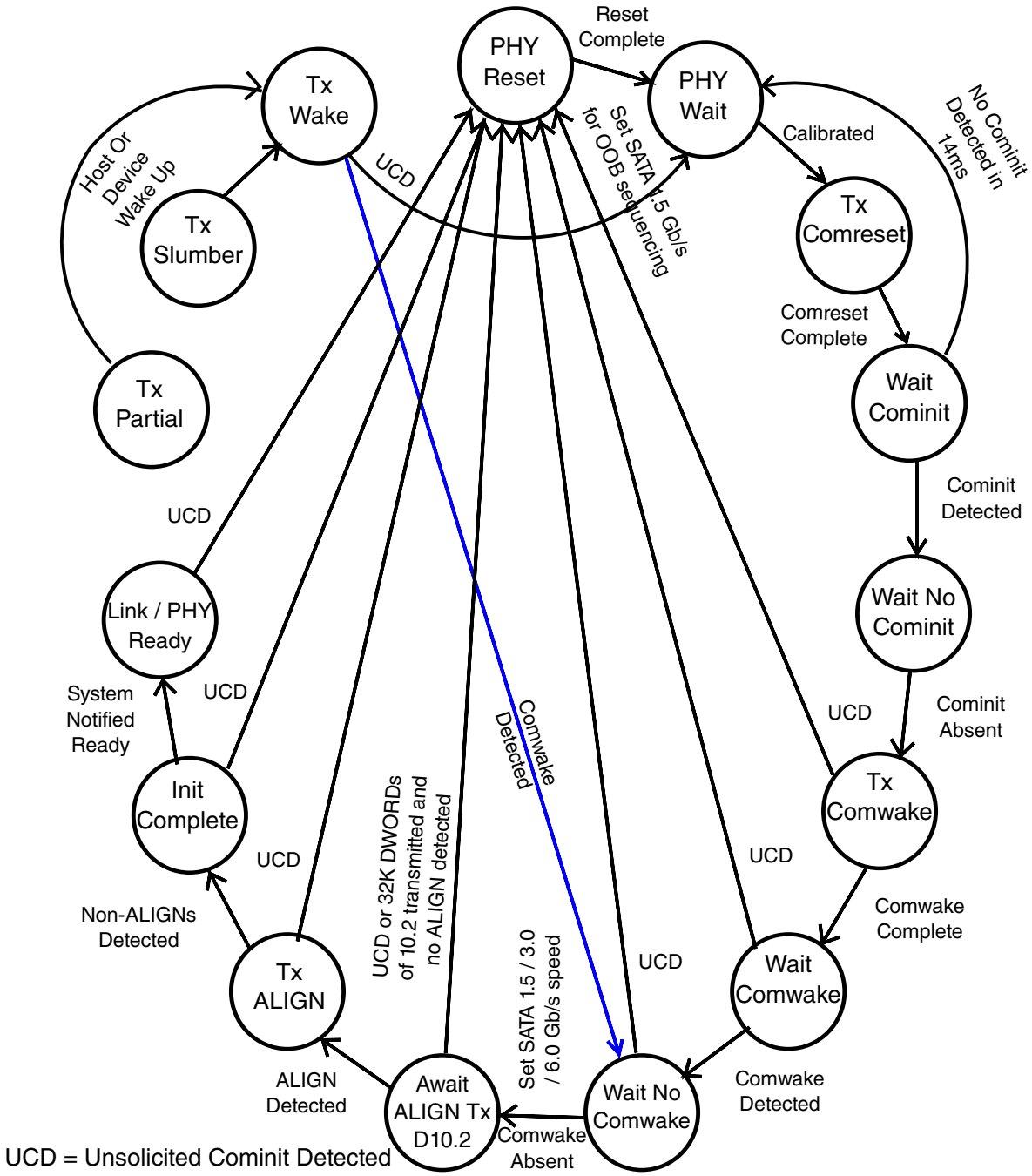


Figure 53-11. Tx OOB Initialization Sequence

53.3.4.5.3.2 Link Layer Tx OOB Sequence Generation

When the Link Layer generates the Tx OOB sequences , the Tx interface connections are different than when the PHY generates them.

Figure 53-11 depicts a small portion of an initialization phase, which includes the first part of a COMRESET condition, followed by a portion of a COMWAKE. The actual OOB sequences are created by sending ALIGN Primitives in conjunction with de-assertion of the phy_tx_enable signal. This causes the Tx OOB data and NULL sequences required by the SATA PHY specifications. Note that this diagram is not the complete initialization sequence. In a complete initialization sequence, there are multiple bursts of COMRESTs and COMWAKEs, followed by a d10.2 stream, followed by an ALIGN stream and then finally normal data. The phy_tx_data_vld is only used for qualifying data in systems that require it, but is unrelated to the OOB sequencing itself.

NOTE

Even when TX OOB signaling is generated in the PHY, the Link Layer still needs to see two ALIGNs, followed by three non-ALIGN primitives to transition from initialization to normal operation.

This feature allows the Link Layer to transition to normal operation without requiring the two ALIGNs. However, this also requires all data from the PHY to be properly aligned and to stay that way. The Link layer does still require three non-ALIGN Primitives, so they must be passed to the core before the incoming data turns into CONT 'Repeat Primitive Data'. This feature does not work when ALIGN_MODE = Misaligned. For more details, see BISTCR.QPHYINIT (bit 14) on page 162.

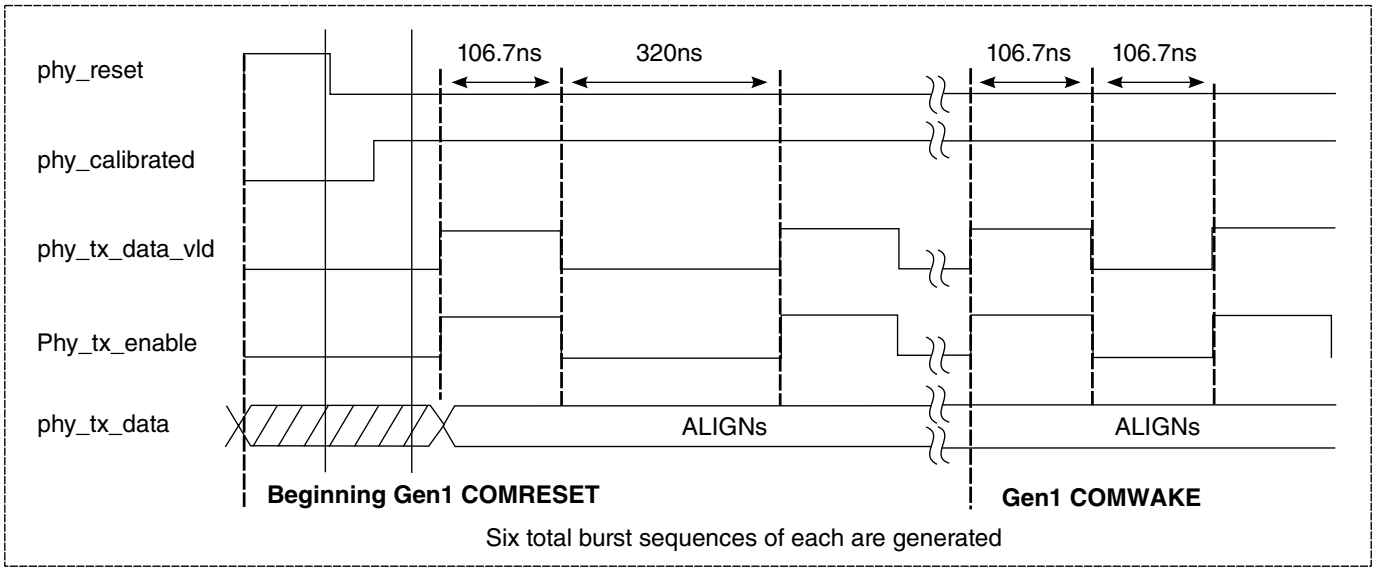


Figure 53-12. Link Layer Generated Tx OOB Signaling

53.3.4.5.3.3 Link Layer Rx OOB Sequence Detection

When the Link Layer detects the Rx OOB sequences, the Rx interface connections are different than when the PHY detects them.

Figure 53-12 depicts a small portion of an initialization phase, which includes the first part of a COMINIT condition, followed by a portion of a COMWAKE.

The actual OOB sequences are comprised of specifically timed ALIGN Primitives in combination with negation of the phy_sig_det signal (indicating NULL on the differential pair). However, only the phy_sig_det signal is used for qualifying Rx OOB sequences by the Link Layer.

NOTE

The PHY must “condition” (filter and delay) phy_sig_det so that the OOB detector accurately detects SATA sequences and prevents the Link Layer from accidentally misreading the Device as disconnected, thus avoiding reset. A filter is required to prevent phy_sig_det from detecting transient ‘loss of signal’ due to bits on the Rxp/Rxn wires crossing the zero voltage level.

To ensure that power modes function properly, phy_sig_det should be timed to the data delay through the PHY so that it does not go low until at least four PMACKs are passed

completely to the core. This delay must also account for the existence of an ‘elasticity buffer’ located in the PHY. Both the filtering and the delay must be ‘symmetric’ (phy_sig_det rise filter/delay equal to the fall filter/delay) so that errors are not introduced in OOB signal detection or impact OOB compliance.

Figure 53-12 does not show the complete initialization sequence. In a complete initialization sequence, the host receives multiple bursts of COMINITs and COMWAKEs, followed by an ALIGN stream, and then finally normal data.

The Link Layer qualifies valid COMINIT and COMWAKE sequences according to the SATA specifications, using calculated comparator values that always fall within specification-compliant ranges. The SATA- specified valid ranges of sequence spacing are shown in Figure 53-12. The actual COMINIT and COMWAKE sequences are calculated and qualified by the Link Layer as described in the following subsections. See the application note and "Example Calculated COMWAKE, COMINIT and Data Burst Lengths" for important information.

To accurately qualify OOB spacing inside the compliant range required by the SATA specifications, the OOB sampling clock, clk_rxoob, must be no lower than 60 MHz.

The phy_rx_data_vld input is used only to qualify data in systems that require it, but is unrelated to the OOB sequencing itself. Systems with this configuration that do not use phy_rx_data_vld (data valid every clock cycle), should tie the bits HIGH. This does not affect OOB detection, because data is qualified with phy_sig_det for this purpose.

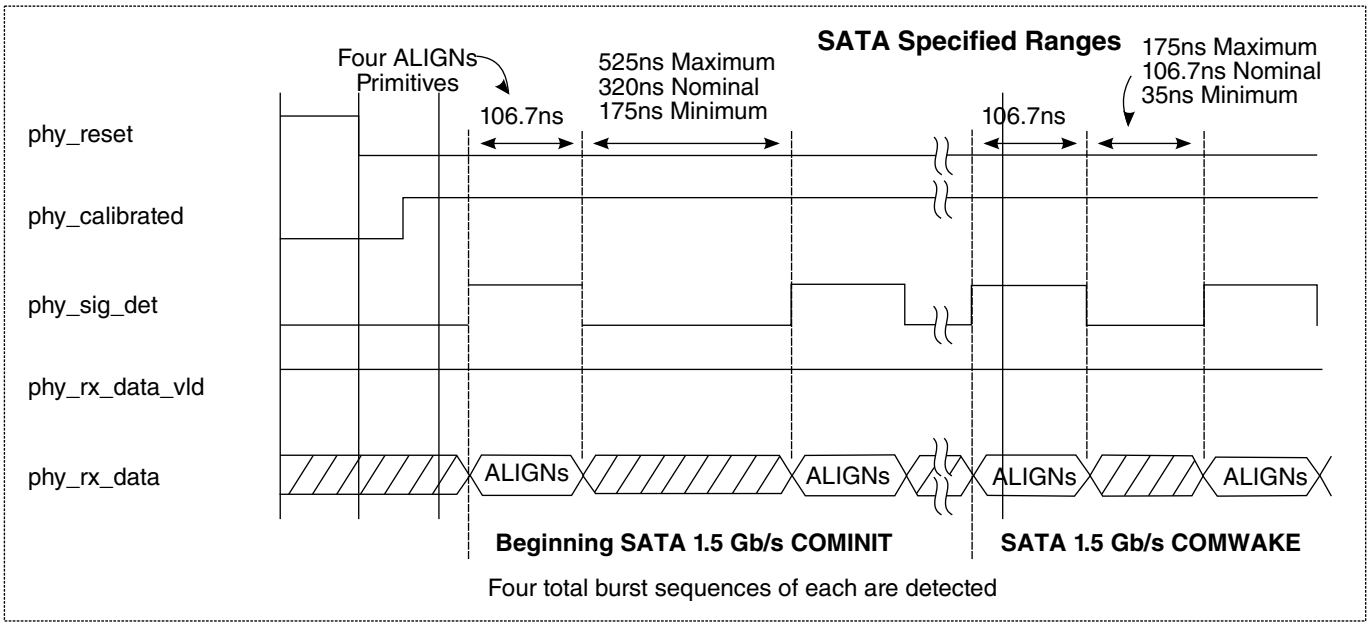


Figure 53-13. Link Layer Rx OOB Detection

53.3.4.5.4 Link Layer Power Management Details

Power management OOB detection and generation sequences occur in the same signaling format as described in PHY Initialization Details. However, they follow the required sequencing specified in the SATA Power management specifications.

In the SATA block, all data on the Tx channel is disabled until the host or device requests a wake-up condition. In addition, Rx data is not allowed to propagate in the SATA block Rx data path until one of these conditions has been detected and the PHY and device have been fully initialized per requirements.

A power down and wake up is depicted in Figure 53-14 and Figure 53-15, excluding actual OOB signaling, which are dependent on OOB Modes. Note that 'OOB' on Tx and Rx Data indicates that OOB sequences are present.

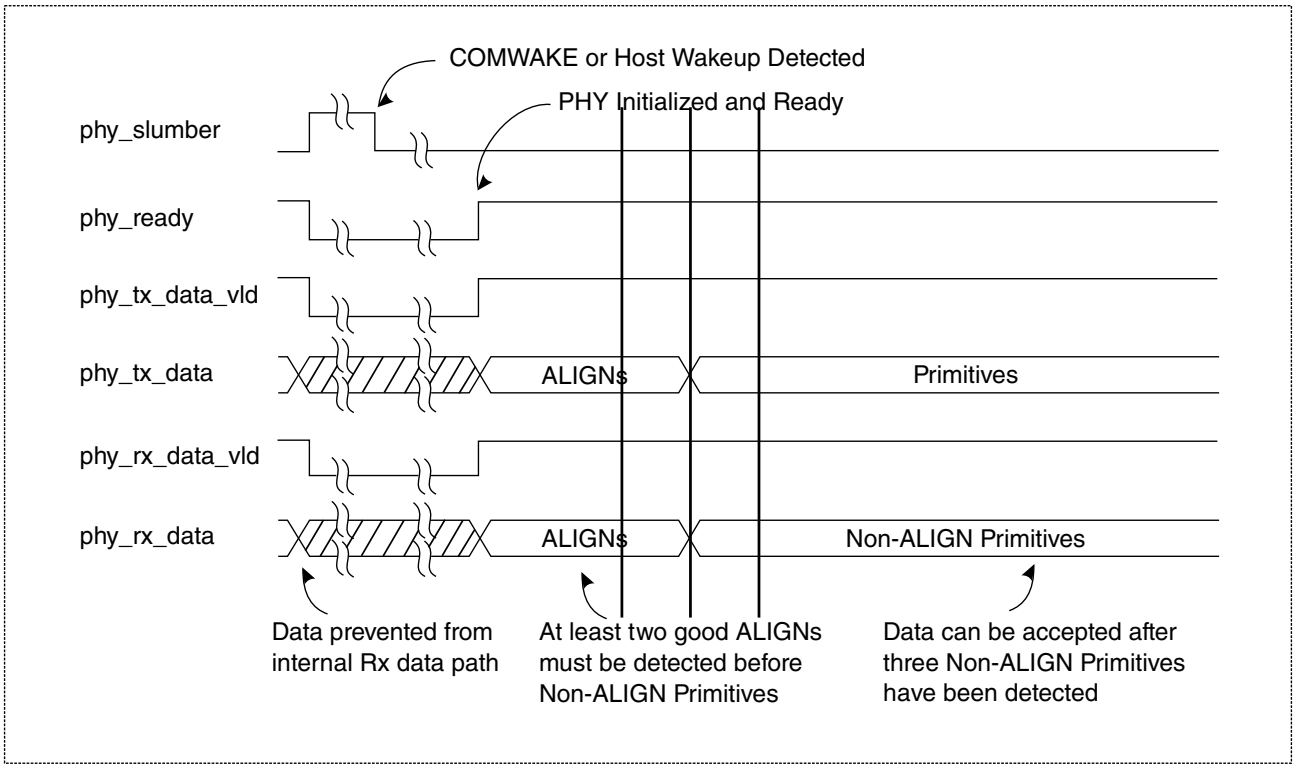


Figure 53-14. Power Mode Example: Rx and Tx OOB In PHY

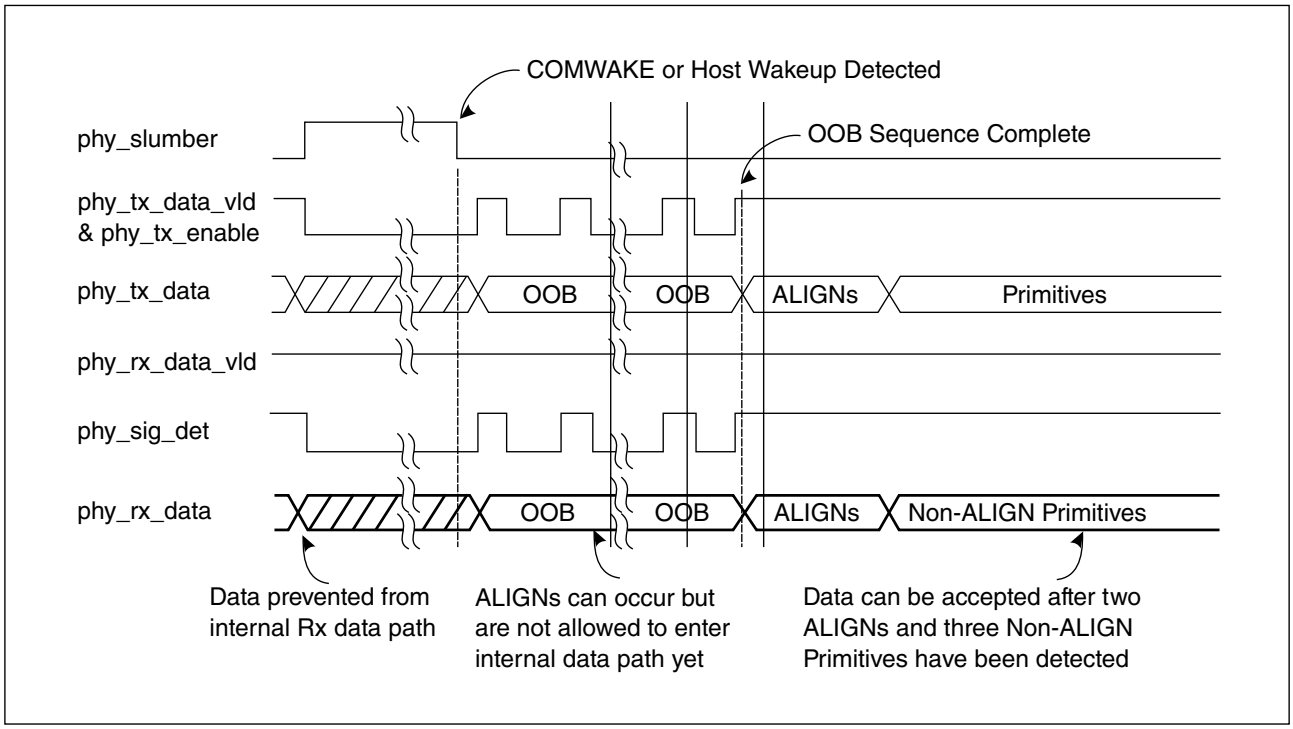


Figure 53-15. Power Mode Example: Rx and Tx In Link

53.3.4.6 Port Power Control Module

The Port Power Control Module (PCM) implements the following functions:

- Monitors Transport, Link and PHY ready/not ready conditions, as well as Device and Host power requests.
- Systematically controls the Link and Transport Layer transitions into and out of offline conditions (system reset, COMRESET and power modes).
- Allows clk_asic0 and clk_rbc0 to be stopped during Slumber and Partial power modes.

The PCM main function is to allow disabling clk_asic0 and clk_rbc0 in SATA power down modes.

CAUTION

Clocks supplied to the core should never glitch at any time, including before, during, and after initialization and power modes. Clock glitch protection should be performed outside the core for any clocks that might glitch.

In addition, inputs to the core should remain static during the time external logic is removing external clock glitches.

For more information, refer to the SATA specifications related to power mode exit time requirements, or contact Synopsys support.

NOTE

If clk_asic0 or clk_rbc0 are stopped, Near End Analog Loopback mode is not supported when a device is connected to the system. Therefore, it is recommended to only stop clocks in Slumber mode, in order to support Near End Analog Loopback mode when a device is connected.

NOTE

In order to support Host-initiated power modes where clk_rbc and clk_asic are removed, the PMACK received from the Device must be able to make it through the clk_rbc clock domain, synchronization, and the Link Layer clk_asic domain Rx Data path to the Link state machine, before the clocks can be removed. The SATA specifications allow a Device to transmit 4 to 16 PMACKs before going into power down. While 16 PMACKs are enough to guarantee receipt by the Link state machine, 4 are not. In the cases where a Device does not send enough PMACKs, the clocks will need to be kept running long enough for the Link state machine to detect the PMACK, or the Host will not go completely into power down mode and a Host COMRESET would be required to exit the failed power mode.

Figure 53-16 shows a high-level state diagram of the Power Control Module. Some signals are omitted for clarity.

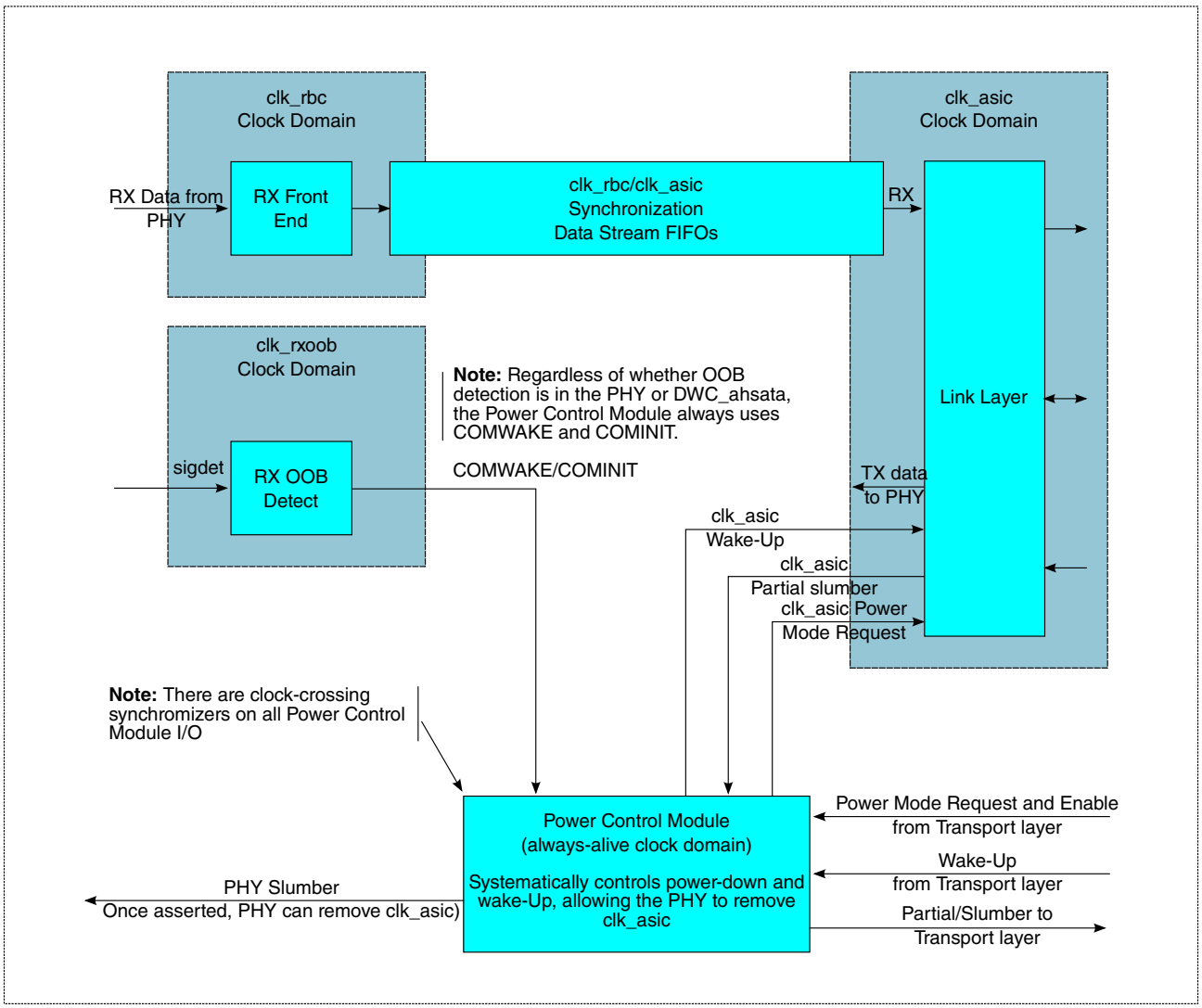


Figure 53-16. Port Power Control Module Diagram

The Power Control Module exists in the 'always alive' clk_pmalive clock domain. The clk_pmalive must always be present and must never change frequency. All signals into and out of the PCM are synchronized between the clk_pmalive, clk_asic0, clk_rbc0, and hclk clock domains with one or more of the synchronizers described in "Cross Clock Synchronization". The Power Control Module serves to assure all SATA block Layers and the PHY move correctly between inactive and active states in unison.

NOTE

Within the core there is no difference between going into and out of Partial and Slumber power modes, even if a system

disables clk_asic0 and clk_rbc0 in one mode, but not the other. Clocks do not have to be removed in either mode.

An example Slumber Power Mode Control with clocks stopped follows:

Example 1: Host Initiated Slumber Power Mode 1

1. System host requests Slumber Power Mode and asserts a Slumber request to the Power Control Module, which then forwards the request to the Link Layer.
2. If the Link Layer was in an IDLE state, then the Link Layer sends PMREQ_Sp primitives to the device. If the device responds with PMACKp primitives, the Link Layer asserts a Slumber signal back to the Power Control Module. If the Link Layer was not IDLE (or already in a power state), or the device denies the Slumber request, the Link Layer sends a power mode abort signal to the Power Control Module, which forwards it to the Transport Layer, and normal operation continues.
3. Provided the Link did not abort the Slumber request, and once the power control module receives sees the Slumber signal from the Link Layer, phy_slumber is asserted to the PHY and clk_asic and clk_rbc can be stopped.
4. At the same time, a slumber signal is sent to the Transport Layer; if no activity or wake up has been detected since the original request was made, the Transport Layer asserts an acknowledge signal back to the power control module. If activity has been detected in the Transport Layer, it responds with a wake up signal back to the power control module and the power mode is cancelled to the PHY. If the Transport Layer allowed the power mode to take place, it records the power down status in register SATA_P 0SSTS[IPM]. Note that the power mode to the PHY must be asserted if the Link Layer had accepted it, given the Tx Data will have been stopped at that point. Therefore, to avoid a power mode assertion to the PHY that is too short, the phy_slumber will be asserted for a minimum of 16 clk_pmalive clock cycles before the signal is asserted to the Transport Layer. This guarantees the power mode is not cancelled before a minimum of 16 cycles of assertion.
5. Once the PHY sees the Slumber signal, p0_phy_slumber, then clk_asic0 and clk_rbc0 can be stopped.

Example 2: Wake Up From Slumber Power Mode

1. The Slumber power mode is stopped by either detection of a COMRESET/ COMWAKE from the device, or software cancelling the power mode. When software has cancelled the power mode, an internal wake up signal is sent from the Transport Layer to the Power Control Module. Whenever the PCM detects the a host "wake up", or a COMRESET or COMWAKE, the Slumber signal to the PHY p0_phy_slumber, is immediately negated, which should result in the PHY restarting clk_asic0 and clk_rbc.

2. At the same time the Slumber is negated to the PHY, Slumber is negated to the Transport Layer and all power requests to the Link Layer are cancelled and normal operation resumes. Because the Link

Layer uses rising edge detection on power requests, and the Transport Layer uses rising edge detects on power mode assertion, unwanted power modes do not occur.

53.3.5 Operation Details

53.3.5.1 Data Transfer

The DMA engine for PORT0 is implemented in the PDMA module. Its operation is based on the AHCI specification Port State Machine.

The following sections outline examples of the ATA DMA read and write transfers.

NOTE

Optional prefetching of ATAPI commands, PRD entries or data, as indicated by the 'P'-bit in the Command Header, is not supported in the current release.

53.3.5.1.1 ATA DMA Read

This is the DMA read sequence:

1. Software finds a free command slot by reading the SATA_P 0CI register, then builds a DMA read command in the Command List for the Port to execute, and sets the bit corresponding to this command slot in the SATA_P 0CI register.
2. The PDMA fetches the Command Header from system memory.
3. The PDMA fetches the command Register FIS from system memory and transfers it to the device.
4. Since this was a DMA read command, the device responds with a number of Data FISes. When they arrive, PDMA performs the following operations:
   - Fetches the first PRD from system memory.
   - Transfers data from the RxFIFO to system memory until the byte count for this PRD is satisfied.
   - Continues to fetch PRDs and transfer data until the byte count for the command is satisfied.
5. Device sends D2H Register FIS with the command ending status and when the I-bit is set, interrupt is generated. D2H Register FIS is posted to the Received FIS memory structure.

6. When this is the last command, and the SATA block was enabled for aggressive power management, the PDMA requests the Link Layer to enter either Partial or Slumber state.

53.3.5.1.2 ATA DMA Write

This is the DMA write sequence:

1. Software finds a free command slot by reading the SATA_P0CI register, then builds a DMA write command in the Command List for the Port to execute and sets the bit corresponding to this command slot in the SATA_P0CI register.
2. The PDMA fetches the Command Header from system memory.
3. The PDMA fetches the command Register FIS from system memory and transfers it to the device.
4. Device responds with DMA Activate FIS. When it arrives, PDMA performs the following operations:
   - Fetches the first PRD from system memory;
   - Transfers data from system memory to TxFIFO until the PRD byte count is satisfied or 8-KB FIS boundary is reached. The Link layer sends Data FIS to the device. If more than one FIS is needed, device sends DMA Activate FIS for each Data FIS;
   - Continues to fetch PRDs and transfer data until the byte count for the command is satisfied.
5. Device sends D2H Register FIS with the command ending status and when the I-bit is set, interrupt is generated. D2H Register FIS is posted to the Received FIS memory structure.
6. When this is the last command, and the SATA block was enabled for aggressive power management, the PDMA requests the Link Layer to enter either Partial or Slumber state.

53.3.5.1.3 Native Queued Command (NCQ) Transfers

The SATA block supports NCQ feature (READ/WRITE FPDMA QUEUED commands). Data transfers are activated via the DMA Setup FIS, and command completion is performed via the Set Device Bits FIS.

NOTE

- Device must also support NCQ, otherwise it aborts READ/ WRITE FPDMA QUEUED commands. The non-zero buffer offset feature should be disabled in the device.
- ATA/ATAPI-7 queued feature set (legacy queueing) is not supported by the AHCI spec.

53.3.5.1.4 PIO Transfer

The SATA block supports multiple DRQ block PIO operation (SATA_CAP[PMD]=1).

From the SATA block point of view, PIO transfer looks like a DMA transfer: a command table is set up, and the data is transferred from or to system memory by the PDMA module.

53.3.5.1.5 Transaction Size

SATA BIU sub-block generates corresponding bus cycles (burst/single read or write) based on the PDMA transaction request. The transaction size is set by the P0DMACR register using RXTS and TXTS fields for SATA receive/write and SATA transmit/read operation, respectively. The RXTS/TXTS values set the corresponding FIFO levels (in FIFO or bus-wide words): RX FIFO ae_level_d (almost empty-level destination) and TX FIFO af_level_s (almost full-level source).

PDMA uses the corresponding FIFO flags: RX FIFO almost_empty_d and TX FIFO almost_full_s to generate DMA requests to the BIU. In some cases, the transaction size can be smaller than the RXTS/TXTS value due to various boundaries that can not be crossed:

- 1KB address boundary for AHB bus
- 4KB address or bus size boundary for AXI bus
- Data FIS boundary
- PRD boundary
- width boundary (if transaction starts with non-bus-aligned address)

Software can change RXTS/TXTS values within the limits specified later in this databook. Generally, the maximum transaction size is half the FIFO depth, except for RX FIFO when the capacity is 64 DWORDs, it is limited to the smaller value due to the af_level_s (almost full level source), which is used to prevent RX FIFO overflow due to the HOLD-HOLDA \ latency. Also, for the AXI bus, RXTS/TXTS values can be limited by the CC_MSTR_BURST_LEN parameter. RXTS/TXTS values are set to the maximum size on power up or asynchronous reset.

DWC_ahsata bus side performance is determined mostly by these factors:

- Bus speed (hclk/aclk frequency)
- RX/TX FIFO size
- Transaction/burst size
- Number of Ports
- Number of other masters (bus loading)

53.3.5.2 Power Management Operations

The SATA block Port power management states (PARTIAL or SLUMBER) can be initiated by the software, the Port itself, or by the device.

The power state machine is implemented in the Link Layer power management module (refer to Link Layer Power Management Details for details). It asserts corresponding signal (p0_phy_partial or p0_phy_slumber) to the PHY to enter the power management state.

Software requests transition to either PARTIAL or SLUMBER state using the SATA_P0CMD[ICC] field, however, the Port acts on it when the Link Layer is currently in the L_IDLE state, otherwise this request is ignored.

The device requests power management state by transmitting PMREQ_Pp or PMREQ_Sp primitives to the Port. Software can disable transition to power management states using the SATA_P0SCTL[IPM field].

The SATA block supports aggressive power management states that allow the Port to initiate an interface power management state as soon as there are no commands outstanding to the device. The SATA_P0CMD[ALPE] bit defines whether the feature is enabled and the SATA_P0CMD[ASP] field controls whether PARTIAL or SLUMBER state is initiated by the Port when enabled. When SATA_P0CMD[ALPE] is set, if the Port recognizes that there are no commands to process, the Port transitions to PARTIAL or SLUMBER state based upon the SATA_P0CMD[ASP] field setting.

The Port recognizes no commands to transmit as either:

- SATA_P0SACT is cleared to 0h, and the SATA_P0CI is updated from a non-zero value to 0h.
- SATA_P0CI is cleared to 0h, and a Set Device Bits FIS is received that updates SATA_P0SACT from a non-zero value to 0h.

SATA block supports automatic PARTIAL to SLUMBER power state transition feature. This is enabled by the software setting the bit SATA_P0CMD[APSTE]. When SATA_P0CMD[APSTE] is set and the SATA block Link is in PARTIAL state, the core automatically transitions to SLUMBER state regardless whether it was host software-, Port (aggressive)-, or device-initiated.

The power management state is terminated when either one of the following conditions becomes true:

- Software requests transition to active state by writing SATA_P0CMD[ICC] = 1h
- Device requests interface wakeup by transmitting COMWAKE OOB sequence

The state of the interface (active, PARTIAL, or SLUMBER power management) is reflected in the SATA_P0SSTS[IPM] field.

53.3.5.3 Hot Plug

This topic covers the following descriptions:

- Native Hot Plug

53.3.5.3.1 Native Hot Plug

The SATA block supports native SATA hot-plug through the use of the SATA_P#0SERR[DIAG_X] and SATA_P#0IS[PCS] bits. It is set every time the Link detects a COMINIT sequence, indicating hot plug insertion event or system power-up.

Hot plug removal is detected by a change in the state of the Port internal "PHY READY" signal. This event is reflected in the SATA_P#0SERR[DIAG_N] and SATA_P#0IS[PRCS] bits.

53.3.5.4 Port Multiplier Support

The SATA block supports Port Multiplier functionality with command-based switching. When a Port is connected to a Port Multiplier, software must first enumerate it by issuing software reset to Port 0Fh (control Port) on the Port Multiplier.

When the signature returned corresponds to a Port Multiplier, then a Port Multiplier is attached. When the signature returned corresponds to another device type, then a Port Multiplier is not attached.

The SATA block provides command list override feature (as indicated by the SATA_CAP[SCLO] = 1) via SATA_P 0CMD[CLO] to help software reliably enumerate the Port Multiplier:

1. Software ensures that SATA_P 0CMD[ST] bit is '0';
2. Software constructs the two Register FISes required for a software reset in the command list, where the PM Port field value in the Register FIS is cleared to 0Fh;
3. Software sets SATA_P 0CMD[CLO] to '1' to force the BSY and DRQ bits in the SATA_P 0TFD register to be cleared;
4. Software sets SATA_P 0CMD[ST] bit to '1' and set appropriate SATA_P 0CI bits in order to begin execution of the software reset command.

53.3.5.5 Interrupts

The SATA block uses a two-tiered interrupt structure defined in the AHCI specification.

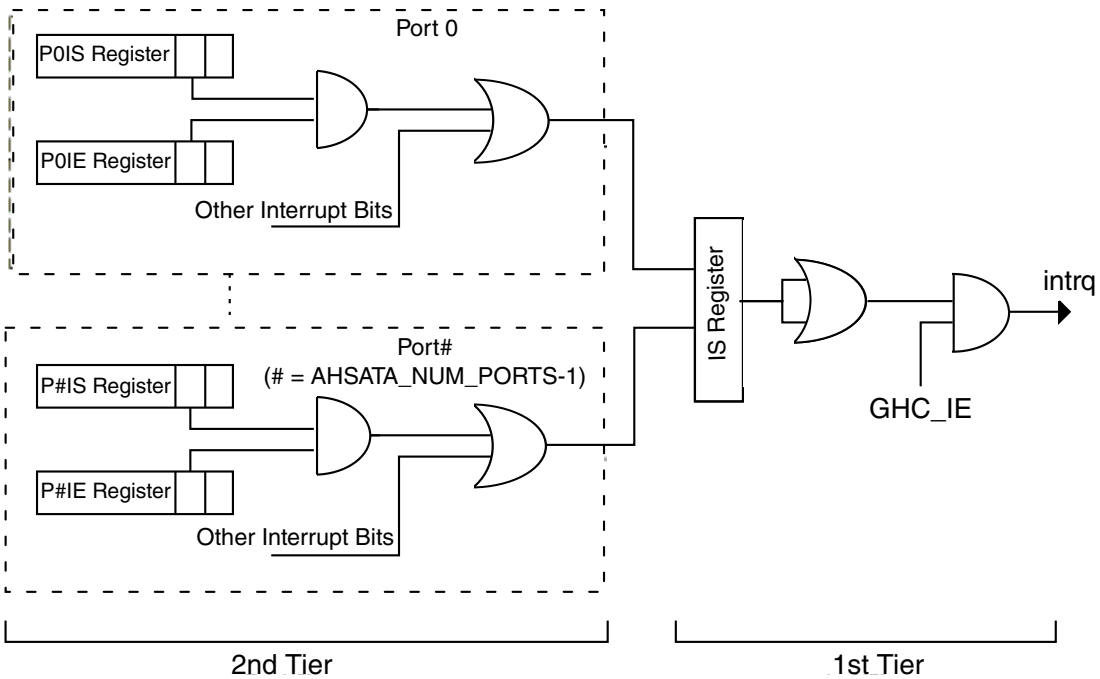


Figure 53-17. SATA block Interrupt Tiers

53.3.5.5.1 First Tier (SATA_IS Register)

The first tier is identified by the IS and SATA_GHC registers. SATA_GHC[IE] bit enables interrupts for the entire SATA block: when it is cleared, intrq output is not asserted regardless of any bits set in the SATA_IS register.

SATA_GHC[IE] bit acts as a mask and does not affect the setting of any interrupt status bits.

The 32-bit SATA_IS register reports the SATA Port has an interrupt pending.

Command Completion Coalescing (CCC) logic generates interrupt (if enabled by software) by setting the IS.IPS[INT] bit, where INT= AHSATA_NUM_PORTS. For example, if DWC_ahsata is configured with eight ports (AHSATA_NUM_PORTS = 8), then the ports use IS bits [7:0], while the CCC interrupt uses IS bit 8.

53.3.5.5.2 Second Tier (SATA_P 0IS Registers)

The second tier is identified in each Port through the SATA_P 0IS (status) and SATA_P 0IE (interrupt enable) register. The SATA_P 0IS register has various interrupt bits that can be individually enabled or disabled by setting the corresponding bit in the SATA_P0IE.

The status bit in the SATA_P 0IS is always set regardless of the setting of the corresponding SATA_P 0IE bit.

53.3.5.5.3 Message Signaled Interrupt

SATA enerates a single message signaled interrupt (MSI) request as one clock period pulse on the msi_req output when the following condition is true:

- IS register changes its state
- IS register is not zero; and
- register GHC.IE=1.

For example, msi_req is asserted when one or more IS bits are set. The signal is also asserted when multiple IS bits are set; some, but not all, bits are cleared; and some other bits have not been set.

This feature is used in PCI-based systems as alternative to the pin-based INT* interrupt.

CAUTION

Because both intrq and msi_req outputs are asserted at the same time, the external logic must disable/gate intrq when the DWC_ahsata is used in the PCI system and when MSI capability is enabled.

The figure below shows an example of connecting SATA interrupts to the PCIE core. Both the intrq and msi_req outputs must be synchronized from the hclk to the PCIE core_clk clock domain (intrq: level sync, msi_req: pulse sync).

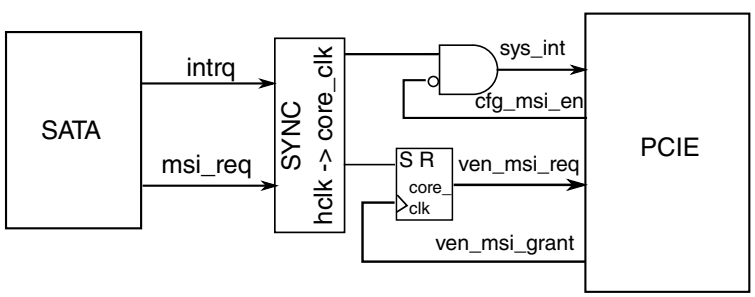


Figure 53-18. Connecting SATA interrupts to PCIE

53.3.5.6 PHY and Link Control

The SATA block provides two registers - SATA_P0PHYCR and SATA_P0PHYSR for PHY control and status respectively (both are located in the Port PCSR module).

SATA_P0PHYCR register is mapped to the corresponding bits of the p0_phy_ctrl output bus and the SATA_P0PHYSR register to the phy_status input bus.

The Port Link Layer features (scrambler, descrambler, repeat drop) are controlled by the SATA_BISTCR[LLC] field (SATA_BISTCR is located in the GCSR module) and can be disabled in normal operation, such as for testing purposes, by clearing the corresponding bits. The required Port is selected using SATA_TESTR[PSEL] register.

SATA_BISTCR[LLC] bits are set on power up enabling scrambler, descrambler, RPD functions by default. To disable these functions, software must:

1. set SATA_P0SCTL[DET]
2. clear the required SATA_BISTCR[LLC] bits
3. clear SATA_P0SCTL[DET]

53.3.5.7 Reset Conditions

53.3.5.7.1 System Reset

System bus resets SATA block by asserting hresetn=0 (asynchronous bus reset). It is usually initiated on power-up or during system bus failure. All components of the SATA block are initialized, including Ports, Generic registers, BIU.

53.3.5.8 Global Reset

Software may globally reset SATA block by setting SATA_GHC[HR]. When software sets the SATA_GHC[HR] bit, the SATA block performs an internal reset action, then clears this bit when the reset is complete.

Writing 0 to SATA_GHC[HR] has no effect.

NOTE

This reset clears the field SATA_P 0SCTL[SPD]. All Port communication is restarted with the maximum allowable speed, as set by the p0_phy_spdmode input.

To perform the Global reset, software sets SATA_GHC[HR] to '1' and may poll until this bit is read to be '0', indicating the reset completion. The SATA block is initialized as follows:

- SATA_GHC[AE], SATA_GHC[IE], the SATA_IS register, and all Port register fields (except SATA_P 0FB and SATA_P 0CLB) that are not HwInit in the register memory space are reset;
- All other global registers/bits and any HwInit bits in the Port-specific registers are not affected by setting SATA_GHC[HR] to '1';
- The Port-specific registers SATA_P 0FB, and SATA_P 0CLB are not affected by setting SATA_GHC[HR] to '1';
- SATA_P 0CMD[SUD] bit is reset to '0'; software is responsible for setting the SATA_P 0CMD[SUD] and SATA_P 0SCTL[DET] fields appropriately such that communication can be established on the SATA link.

53.3.5.8.1 Port Reset (COMRESET)

Software causes a Port reset by writing 1h to the SATA_P 0SCTL[DET] field to invoke COMRESET on the interface and start a re-establishment of the PHY Layer communication.

Software should wait at least 1ms before clearing SATA_P 0SCTL[DET] to 0h; this ensures that at least one COMRESET signal is sent over the interface. After clearing SATA_P 0SCTL[DET] to 0h, software should wait for communication to be re-established as indicated by bit 0 of SATA_P 0SSTS[DET] being set to '1'. Then software should write all ones to the SATA_P 0SERR register to clear any bits that were set as part of the Port reset.

NOTE

SATA asserts the corresponding p 0_phy_reset(_n) output when P 0SCTL.DET=0x1. It negates p 0_phy_reset(_n) and sends a COMRESET OOB sequence when P 0SCTL.DET=0x0.

53.3.5.8.2 Software Reset

Software builds two H2D Register FISes in the command list.

The first Register FIS has the SRST bit set to '1' in the Control field of the Register FIS, the 'C' bit is cleared to '0' in the Register FIS, and the command table has the CH[R] (reset) and CH[C] (clear BSY on R_OK) bits set to '1'. The CH[R] (reset) bit causes the Port to perform a SYNC escape when necessary to put the device into an idle condition before sending the software reset. The CH[C] (clear BSY on R_OK) bit needs to be set for the first Register FIS to clear the BSY bit and proceed to issue the next Register FIS

since the device does not send a response to the first Register FIS in a software reset sequence. The second Register FIS has SRST='0' in the Control field of the Register FIS, the 'C' bit is cleared to '0'in the Register FIS, and the command table has the CH[R] (reset) and CH[C] (clear BSY on R_OK) bits cleared to '0'. When issuing a software reset sequence, there should not be other commands in the command list. Before issuing the software reset, software must clear SATA_P 0CMD[ST], wait for the Port to be idle (SATA_P 0CMD[CR]='0'), and then set SATA_P 0CMD[ST] bit again. SATA_P 0TFD[STS] BSY bit and SATA_P 0TFD[STS] DRQ bit must be cleared prior to issuing the reset. When SATA_P 0TFD[STS] BSY bit or the SATA_P 0TFD[STS] DRQ bit is still set based on the failed command, then a Port reset should be attempted or command list override (SATA_P 0CMD[CLO]) should be used.

NOTE

A Port reset (COMRESET) is the preferred mechanism for error recovery and should be used in place of software reset.

53.3.5.9 Interface Speed Support

The SATA block supports 1.5 Gb/s and 3 Gb/s interface speeds as indicated by the SATA_CAP[ISS]=2h. Software can limit a Port's speed to 1.5 Gb/s by setting SATA_P 0SCTL[SPD] field to 1h.

53.3.5.10 Staggered Spin-up

The SATA block supports staggered spin-up operation when SATA_CAP[SSS]='1'. This feature is used to individually spin-up attached devices, thus reducing power supply requirements for multiple devices power-up.

NOTE

In order for a system to support staggered spin-up, the devices and BIOS/driver software must also support staggered spin-up.

The SATA_P 0CMD[SUD] bit is used to manipulate the PHY behavior. SATA_P 0SCTL[DET] and SATA_P 0CMD[SUD] must be set correctly in order to avoid illegal combinations of the two values.

The following table describes interaction between the SATA_P 0CMD[SUD] and SATA_P 0SCTL[DET] bits.

Table 53-3. SATA_P 0CMD[SUD] and SATA_P 0SCTL[DET] Interaction

SATA_P0SCTL[DET]	SATA_P0CMD[SUD]	Mode	Behavior
0h	0	Listen	Interface is in a reduced power state. When COMINIT is received then SATA_P 0SERR[DIAG_X] is set and no response (OOB signal) is sent to the device COMWAKE is ignored. The applcation must place the Port into this state only when no device is detected as connected to this Port. In this mode, the Port forces the PHY into a low power state without requesting a SLUMBER transition on the link.
0h	0 -> 1	Spin-Up	Port sends COMRESET, begins initialization sequence.
0h	1	Normal	Normal operating state when the Port is performing data transfers.
1h	0	(not allowed)	This combination is prohibited in hardware. SATA_P 0CMD[SUD] can not be cleared when SATA_P0 SCTL[DET]=1h, and SATA_P0 SCTL[DET] can not be set to 1h when SATA_P0 CMD[SUD}=0.
1h	1	Reset	Port continuously transmits COMRESET and does not listen for COMINIT. When COMINIT is received in this state, the SATA_P0 SERR[DIAG_X] bit is set.
1h-> 0h	1	Initialize	Port stops sending COMRESET, begins initialization sequence.
4h	N/A	Off	Port PHY is off.

Software must only clear SATA_P 0CMD[SUD] when it believes that no device is attached. In Listen Mode (SATA_P 0SCTL[DET]=0h and SATA_P 0CMD[SUD]=0), the Port PHY enters a reduced power state, equivalent to the SLUMBER power management state. The Port PHY enters this state without negotiating a transition to SLUMBER on the link, as asking for a transition to SLUMBER when no device is attached fails, and therefore the PHY remains in a high power state. To avoid this software should ensure that SATA_P 0SSTS[DET]=0h indicating that no device is present before clearing SATA_P 0CMD[SUD].

In order to spin up the devices attached to the SATA block Ports, software should perform the procedure outlined in Firmware Specific Initialization.

53.3.5.11 Activity LED

The SATA_CAP[SAL]=1 indicates that the activity LED feature is enabled to software.

P0_act_led output is used to drive an external LED based upon activity of the Port:

- '1' - LED On (Port active)
- '0' - LED Off (Port inactive)

The Port drives the LED active (p0_act_led='1') if:

- (SATA_P 0CI != 0h or SATA_P 0SACT != 0h) and SATA_P 0CMD[ATAPI]= '0';
- (SATA_P 0CI != 0h or SATA_P 0SACT != 0h) and SATA_P 0CMD[ATAPI] = '1' and SATA_P 0CMD[DLAE] = '1'.

The Port drives the LED off (p0_act_led='0') when SATA_P 0CI and SATA_P 0SACT are both cleared to 0h.

53.3.5.12 Asynchronous Notification

The SATA block supports asynchronous notification feature as indicated by the SATA_CAP[SSNT]=1. This feature allows an ATAPI device to send a signal to the host when media is inserted or removed and avoids polling the device for media changes.

The signal sent to the host is a Set Device Bits FIS with the 'I' (interrupt) and'N' (notification) bits set to '1'.

To use asynchronous notification, software should set the SATA_P 0IS[SDBS] bit to enable interrupt notification on a Set Device Bits FIS. When accesses to the ATAPI device are idle, software should place the device in a low power state. When the device has a media change, it signals this to the block's SATA Port with a Set Device Bits FIS. In response to receiving a P 0IS[SDBS] interrupt on an idle Port, software should interrogate the device to determine the cause of the interrupt.

The first DWORD of any FIS received by the host contains a 4-bit Port Multiplier Port (PM Port) field. The second DWORD of the BIST Activate FIS least significant byte (bits [7:0]) is stored in the SATA_BISTAFR[NCP] field. The PM Port field indicates which Port/target behind the Port Multiplier issued the FIS to the block's SATA Port. When a Set Device Bits FIS is received by the block's SATA Port and the 'N' (notification) bit is set, the bit position in the SATA_P 0SNTF register corresponding to the PM Port field is set. The block's SATA Port sets the SATA_P 0IS[SDBS] field when the 'I' (interrupt) bit is set in the Set Device Bits FIS. This causes an interrupt to be generated when that interrupt is enabled.

NOTE

When a Port Multiplier is not present, the PM Port field in the Set Device Bits FIS is 0h, causing bit 0 of the SATA_P 0SNTF register to be set.

53.3.5.13 BIST Operation

The SATA Port can be put into one of the BIST loopback modes described in the following subsections.

NOTE

Scrambler and Descrambler are bypassed (disabled) in the Port Link layer in all BIST modes by default. When needed, Scrambler and Descrambler can be enabled through software, by clearing the bits SATA_BISTCR[SCRAM] and SATA_BISTCR[DESCRAM] (SCRAM and DESCRAM are bits within the SATA_BISTCR[LLC] field) prior to entering BIST mode.

53.3.5.14 Loopback Responder

Software must ensure that the Port is in idle state and there are no outstanding commands by checking SATA_P 0CI

and SATA_P 0SACT registers are both cleared, SATA_P 0TFD[STS] register BSY, DRQ and ERR bits are all cleared.

The block's SATA Port enters one of the BIST loopback responder modes when a corresponding BIST Activate FIS is successfully received from the device and is supported by the SATA block. SATA_P 0SSTS[DET] field returns 4h when read. Since BIST registers' locations are shared between all the active Ports, software must first select the Port for BIST operation by writing the Port number to the SATA_TESTR[PSEL] field before accessing SATA_BISTAFR (Port0 is selected in the SATA_TESTR[PSEL] on power-on (system) or global SATA block reset).

NOTE

When the device sends a BIST Activate FIS with a request to enter a non-supported loopback mode, SATA block responds with R_ERRp response upon reception of the FIS.

The following loopback responder modes are supported by the SATA block:

- Far-end retimed
  - The Port receives BIST Activate FIS with Pattern Definition field (bits [23:16] of the first DWORD) = 0x10 from the RxFIFO and stores it in the SATA_BISTAFR[PD] field
  - All the data received from the device in the form of a SATA-compliant pattern is retimed in the Link Layer and transmitted back to the device.
  - Alternately, this mode can be initiated with device disconnected from the Port PHY (Link is in NOCOMM state) when software writes SATA_BISTCR[FERLB]=1. After the device is connected to the SATA block, the device must transmit the number of ALIGNs required for the PHY to sync.
- Far-end analog (Port PHY must support this mode)

- The Port receives BIST Activate FIS with Pattern Definition field (bits [23:16] of the first DWORD) = 0x08 from the RxFIFO and stores it in the SATA_BISTAFR[PD] field
- The Port asserts p0_phy_farafelb signal to the PHY to put it to the Far-end analog loopback mode. The PHY receives and retransmits the raw data without retiming

- Far-end transmit only
  - The Port receives BIST Activate FIS with Pattern Definition field (bits [23:16] of the first DWORD) = 0x80 (scrambling is enabled) or 0xA0 (scrambling is bypassed) from the RxFIFO The Port stores it in the SATA_BISTAFR[PD]. The second DWORD of the BIST Activate FIS least significant byte (bits [7:0]) is stored in the SATA_BISTAFR[NCP] field.
  - The Port transmits corresponding a SATA non-compliant test pattern to the device based on the SATA_BISTAFR[NCP] value:
    - 0xF1: Low transition density pattern (LTDP)
    - 0xB5: High transition density pattern (HTDP)
    - 0xAB: Low frequency spectral component pattern (LFSCP)
    - 0x7F: Simultaneous switching outputs pattern (SSOP)
    - 0x8B: Lone bit pattern (LBP)
    - 0x78: Mid frequency test pattern (MFTP)
    - 0x4A: High frequency test pattern (HFTP)
    - 0x7E: Low frequency test pattern (LFTP)
  - Alternately, this mode can be initiated with device disconnected from the Port PHY (Link NOCOMM state) when software writes a one to the SATA_BISTCR[TXO] bit. SATA block transmits non-compliant BIST pattern defined by the value in the SATA_BISTCR[PATTERN] field.

Loopback responder BIST modes can be exited either when the device signals COMINIT OOB condition, or when the software initiates Port reset (COMRESET).

53.3.5.14.1 Loopback Initiator

The software first selects the Port for BIST operation by writing the Port number to the SATA_TESTR[PSEL] field, then the required pattern by writing to the SATA_BISTCR[PATTERN] field.

The software builds a BIST FIS with the required mode in the commands list and sets CTBAz[B]. Once a BIST command is placed into the list, software is not allowed to build any more commands until it clears SATA_P 0CMD[ST]. After the Port successfully transmits this FIS, it enters this mode and generates/receives the compliant test pattern selected by the SATA_BISTCR[PATTERN] field.

SATA_P 0SSTS[DET] returns 4h when read. SATA_BISTSR and SATA_BISTFCTR registers are updated with error/FIS count information for each received BIST FIS.

NOTE

The device must support either PARTIAL or SLUMBER power modes for near-end analog loopback mode, otherwise it should be initiated with the device disconnected from the Port PHY. It is not clear how the device responds when it does not support the requested BIST mode - with R_OKp and ignoring the request or with R_ERRp. In the former case, the host assumes the device has entered the BIST mode and starts the test that fails.

The following BIST initiator modes can be requested by the software:

- "Far-end retimed"
- "Far-end analog"
- "Near-end analog"
- "Far-end transmit only"

53.3.5.14.1.1 Far-end retimed

The host software writes the SATA_BISTCR[PATTERN] field to select one of the SATA-defined compliant patterns:

- 0000b: Simultaneous switching outputs pattern (SSOP)
- 0001b: High transition density pattern (HTDP)
- 0010b: Low transition density pattern (LTDP)
- 0011b: Low frequency spectral component pattern (LFSCP)
- 0100b: Composite pattern (COMP)
- 0101b: Lone bit pattern (LBP)
- 0110b: Mid frequency test pattern (MFTP)
- 0111b: High frequency test pattern (HFTP)
- 1000b: Low frequency test pattern (LFTP)

The software prepares BIST Activate FIS with bits [23:16]=10h of the first DWORD (Pattern Definition field) in the command list. The Port sends this BIST Activate FIS to the device.

After successful transmission of the BIST Activate FIS (device acknowledges the FIS with R_OKp) the Port generates the requested compliant pattern in the form of BIST frames continuously and checks for errors on the receive side.

SATA_BISTFCTR register is updated with the received BIST frame count and SATA_BISTSR with frame/burst error count. . SATA_P 0SERR register is updated with CRC, disparity, and 10B8B errors for each frame. SATA_BISTFCTR, and SATA_BISTSR registers can be cleared by writing '1' to the SATA_BISTCR[CNTCLR] bit.

To change the pattern, the software issues a Port Reset (COMRESET), writes to the SATA_BISTCR[PATTERN] field to select a new pattern and re-issues the command by setting the SATA_P 0CI bit.

53.3.5.14.1.2 Far-end analog

The software prepares BIST Activate FIS with bits [23:16]=08h of the first DWORD (Pattern Definition field) in the command list. The Port sends this BIST Activate FIS to the device.

The operation proceeds as described in the far-end retimed test above.

53.3.5.14.1.3 Near-end analog

NOTE

Port PHY must support this mode, plus both clk_asic0 and clk_rbc0 must be running in the selected power management states or when the device is disconnected. When this is not true, this loopback mode is not supported by the SATA block.

This mode can be initiated either in one of the power management modes (PARTIAL or SLUMBER) or with the device disconnected from the Port PHY (Link NOCOMM state). The software issues a PARTIAL or SLUMBER power state request to the device via SATA_P 0CMD[ICC] field and sets the SATA_BISTCR[NEALB] bit. The SATA_BISTCR[PATTERN] field selects the required BIST pattern.

The Port asserts p 0_phy_nearafelb to the PHY. The PHY loops the data from its transmitter to its receiver and ignores any data coming from the device.

The operation proceeds as described in the far-end retimed test above, except BIST Activate FIS is not sent to the device.

53.3.5.14.1.4 Far-end transmit only

The software prepares BIST Activate FIS in the command list with the second and third DWORDs containing the required pattern, and the first DWORD - with the Pattern Definition (bits [23:16]) value corresponding to the required mode - Bit 23 (T) is set, bits 20 (L), 19 (F), 17 (R) cleared and bits 22, 21 and 18 are used to enable the following options:

- Bit 22 (A) is set - Bypass ALIGN
- Bit 21 (S) is set - Bypass scrambling
- Bit 18 (P) is set - Primitive bit (refer to the SATA specification for more details)

The Port sends this BIST Activate FIS to the device. After the device acknowledges the reception of this FIS with R_OKp, the Port disables the PHY receiver and transmitter (any received data is ignored by the Link Layer, transmitter is idle and maintains common mode bias per SATA specification).

Loopback initiator BIST modes can be terminated either by the device when it signals COMINIT OOB condition (except the near-end analog mode), or when the software initiates Port reset (COMRESET).

53.3.5.15 Command Completion Coalescing

A Command Completion Coalescing (CCC) feature is used to reduce the interrupt and command completion overhead in a heavily-loaded system.

The SATA block core generates an interrupt to allow software to process completed commands when either of these conditions is true:

- A software-specified number of commands have completed
- A software-specified time-out has expired

This feature applies to all Ports selected to be in the CCC set by software via the SATA_CCC_PORTS register.

CCC logic uses SATA block-specific register SATA_TIMER1MS to generate 1ms interval based on the AHB clock frequency. Software must load this register with the required value before enabling the CCC feature:

- Fhclk * 1000, where Fhclk = AHB clock frequency, in MHz

Additional Command Completion Coalescing details and examples can be found in the AHCI specification.

NOTE

CCC logic can be removed from the SATA core if it is not needed by setting the CCC_SUPPORT configuration parameter to “Exclude” (0). In this case, all CCC-related registers become “reserved” (returns 0 on read).

53.4 Programming

The SATA block software initialization consists of two independent phases: a firmware phase (platform BIOS) and a system software phase.

This section contains the following sub-sections:

- Firmware Specific Initialization
- System software Specific Initialization

53.4.1 Firmware Specific Initialization

The firmware initialization is done on power-up.

The following registers should be initialized to values that reflect the capabilities supported by the platform:

- SATA_CAP[SSS] - support for staggered spin-up
- SATA_CAP[SMPS] - support for mechanical presence switches
- SATA_PI - Ports implemented
- SATA_P 0CMD[HPCP] - whether the Port is hot plug capable. The SATA_P 0CMD[HPCP] should be set to 1 when SATA_P 0CMD[MPSP] or SATA_P 0CMD[CPD] is set to 1 for the Port.
- SATA_P 0CMD[MPSP] - whether mechanical presence switch is attached to the Port.
- SATA_P 0CMD[CPD] - whether cold presence detect logic is attached to the Port.

NOTE

Firmware should initialize the HPCP, MPSP, and CPD bits for each Port implemented on the platform as defined by the SATA_PI register.

After firmware has initialized the above mentioned registers, it should then perform the following steps to complete the staggered spin-up process (when applicable to the platform) on each Port implemented (as indicated by the SATA_PI register):

1. Ensure that SATA_P 0CMD[ST]=0, SATA_P 0CMD[CR]=0, SATA_P 0CMD[FRE]=0, SATA_P 0CMD[FR]=0, and SATA_P 0SCTL[DET]=0.
2. Allocate memory for the command list and the FIS receive area. Set SATA_P 0CLB 0 to the physical address of the allocated command list. Set SATA_P 0FB to the physical address of the allocated FIS receive area. Then set P 0CMD[FRE] to 1.
3. Initiate a spin-up of the SATA drive attached to the Port by setting P 0CMD[SUD] to 1
4. Wait for a positive indication that a device is attached to the Port (the maximum time to wait for presence indication is specified in the Serial ATA specification). This is done by polling SATA_P 0SSTS[DET]. When SATA_P 0SSTS[DET] returns a value of 1h or 3h when read, then the firmware should continue to the next step, otherwise when polling process times out, it moves to the next implemented Port and returns to Step 1.
5. Clear the SATA_P 0SERR register by writing ones to each implemented bit location.
6. Wait for indication that SATA drive is ready. This is determined through examination of SATA_P 0TFD[STS]. When SATA_P 0TFD[STS] BSY bit, SATA_P 0TFD[STS] DRQ bit, and SATA_P 0TFD[STS] ERR bit are all 0, prior to the maximum allowed time as specified in the ATA/ATAPI-7 specification, the device is ready.

53.4.2 System software Specific Initialization

Software may perform the SATA block global reset prior to initializing by setting SATA_GHC[HR] to 1 when desired.

When firmware (BIOS) already allocated memory and initialized the appropriate registers for the command list and FIS receive area, the software may skip this step in the process.

Following is the list of steps for system software to place the SATA block into a minimally initialized state:

1. Determine which Ports are implemented by the SATA block, by reading the PI register. This bit map value aids the software to determine how many Ports are available and which Port registers need to be initialized.
2. Ensure that the SATA block is not in the running state by reading and examining each implemented Port's SATA_P 0CMD register. When SATA_P 0CMD[ST], SATA_P 0CMD[CR], SATA_P 0CMD[FRE] and SATA_P 0CMD[FR] are all cleared, the Port is in an idle state. Otherwise, the Port is not idle and should be placed in the idle state prior to manipulating the SATA block global and Port specific register. System software places a Port into the idle state by clearing SATA_P 0CMD[ST] and waiting for SATA_P 0CMD[CR] to return 0 when read. Software should wait at least 500ms for this to occur. When SATA_P 0CMD[FRE] is set to 1,

software should clear it to 0 and wait at least 500ms for SATA_P 0CMD[FR] to return 0 when read. When SATA_P 0CMD[CR] or SATA_P 0CMD[FR] do not clear to 0 correctly, then software may attempt a Port reset or a global reset to recover.

3. Determine how many command slots the HBA supports, by reading SATA_CAP[NCS].
4. For each implemented Port, system software should allocate memory for and program:
   - SATA_P 0CLB
   - SATA_P 0FB

NOTE

It is good practice for system software to zero-out the memory allocated and referenced by SATA_P 0CLB and SATA_P 0FB. After setting SATA_P 0FB to the physical address of the FIS receive area, system software should set SATA_P 0CMD[FRE] to 1.

5. For each implemented Port, clear the SATA_P 0SERR register, by writing ones to each implemented bit location.
6. Determine which events should cause an interrupt and set the appropriate enable bits of the SATA_P 0IE register. To enable the SATA block to generate interrupts, system software must also set SATA_GHC[IE] to 1.

NOTE

Due to the multi-tiered nature of the SATA block interrupt architecture, system software must always ensure that the SATA_P 0IS (clear this first) and SATA_IS[IPS] (clear this second) register are cleared to '0' before programming the SATA_P 0IE and SATA_GHC[IE] registers. This prevents any residual bits set in these registers from causing an interrupt to be asserted.

Software should not set SATA_P 0CMD[ST] to 1 until it is determined that a functional device is present on the Port as determined by SATA_P 0TFD[STS] BSY bit, SATA_P 0TFD[STS] DRQ bit, and SATA_P 0TFD[STS] ERR bit all cleared, and SATA_P 0SSTS[DET]=3h. To enable the SATA_P 0TFD register to be updated with the initial Register FIS for a Port, the SATA_P 0SERR[DIAG_X] bit must be cleared to 0.

53.5 Software Manipulation of Port DMA

This section contains the following topics:

- Start (SATA_P 0CMD[ST])
- FIS Receive Enable (SATA_P 0CMD[FRE])

53.5.1 Start (SATA_P 0CMD[ST])

When SATA_P 0CMD[ST] is set to 1, software is not allowed to perform the following actions:

- Manipulate SATA_P 0CMD[POD] to power on or off a device through cold presence detect logic (when supported by the platform and enabled in the SATA block);
- Manipulate SATA_P 0SCTL[DET] to change the PHY state;
- Manipulate SATA_P 0CMD[SUD] to spin-up the device (when supported by the platform)

The above actions are only allowed while the Port is in the Not Running state, indicated by both SATA_P 0CMD[ST] and SATA_P 0CMD[CR] being 0.

Software should set SATA_P 0CMD[ST] only after the following conditions become true:

- SATA_P 0CMD[CR] is verified to be cleared to '0' and SATA_P 0CMD[FRE] has been set to 1;
- A functional device is present on the Port (as determined by SATA_P 0TFD[STS] BSY bit=0, SATA_P 0TFD[STS] DRQ bit=0, and SATA_P 0SSTS[DET]=3h) and the registers SATA_P 0CLB are programmed to valid values.

53.5.2 FIS Receive Enable (SATA_P 0CMD[FRE])

When SATA_P 0CMD[FRE] is set (causing SATA_P 0CMD[FR] to be set to 1), the Port receives FISes from the devices and copies them into system memory. When SATA_P 0CMD[FRE] is cleared (causing SATA_P 0CMD[FR] to be cleared to 0), received FISes are held in the RxFIFO, and when it is full, further FIS reception is blocked.

Software is allowed to manipulate SATA_P 0CMD[FRE] so that it may move the FIS receive area to a new location. When this bit is cleared to 0, software must first wait for SATA_P 0CMD[FR] to clear to 0, indicating that the Port DMA engine for FIS reception is in an idle condition. When SATA_P 0CMD[FR] and SATA_P 0CMD[FRE] are both cleared to 0, software may update the values of SATA_P 0FB. Prior to setting SATA_P 0CMD[FRE] to 1, software should ensure that SATA_P 0FB are set to valid values. Software should not write SATA_P 0FB while SATA_P 0CMD[FRE] is set to 1.

Software should set SATA_P 0CMD[FRE] to 1 prior to setting SATA_P 0CMD[ST] to 1. Software should not clear SATA_P 0CMD[FRE] while SATA_P 0CMD[ST] or SATA_P 0CMD[CR] is set to 1.

Upon global or Port reset, the SATA_P 0CMD[FRE] bit is cleared. The D2H Register FIS containing the device signature is accepted by the Port, and the signature field is updated.

NOTE

When the SATA block Port stops running due to an error (e.g., SATA_P 0IS[IFS] is set to 1), FISes may not be posted until the SATA_P 0CMD[ST] bit is cleared to 0 to recover from the error.

53.6 Register Descriptions

This section contains register memory maps, register groups, and bit-field descriptions.

53.6.1 Register Overview

The SATA block registers occupy 328 bytes of the 8-KB address space assigned to the SATA block and are divided into two parts: Generic control, and Port control.

Register space above the offset of 0x14F is reserved.

53.6.1.1 Register Basics:

All registers are 32-bits wide and can be accessed using 8, 16, or 32-bit wide transfers, except when noted otherwise in the register descriptions.

All registers that start below the offset 0x100 are global and apply to the entire SATA block.

53.6.1.2 Reserved Locations

When a register, field, or bit is reserved:

- Reads return zero.
- Writes have no effect.

53.7 SATA Memory Map/Register Definition

This section provides high-level summaries of the Generic and Port control regsier maps.

The register names in SATA Memory Map are cross-referenced to the detailed register descriptions in the following section (double-click on the register name to link to the detailed description).

SATA memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
220_0000	HBA Capabilites Register (SATA_CAP)	32	R	0000_0000h	53.7.1/4552
220_0004	Global HBA Control Register (SATA_GHC)	32	R/W	8000_0000h	53.7.2/4555
220_0008	Interrupt Status Register (SATA_IS)	32	R/W	0000_0000h	53.7.3/4556
220_000C	Ports Implemented Register (SATA_PI)	32	R	See section	53.7.4/4557
220_0010	AHCI Version Register (SATA_VS)	32	R	0001_0300h	53.7.5/4557
220_0014	Command Completion Coalescing Control (SATA_CCC_CTL)	32	R/W	See section	53.7.6/4558
220_0018	Command Completion Coalescing Ports (SATA_CCC_PORTS)	32	R/W	0000_0000h	53.7.7/4559
220_0024	HBA Capabilities Extended Register (SATA_CAP2)	32	R	0000_0004h	53.7.8/4560
220_00A0	BIST Activate FIS Register (SATA_BISTAFR)	32	R	0000_0000h	53.7.9/4561
220_00A4	BIST Control Register (SATA_BISTCR)	32	R/W	0000_0700h	53.7.10/ 4562
220_00A8	BIST FIS Count Register (SATA_BISTFCTR)	32	R	0000_0000h	53.7.11/ 4565
220_00AC	BIST Status Register (SATA_BISTSR)	32	R	0000_0000h	53.7.12/ 4566
220_00BC	OOB Register (SATA_OOBR)	32	R/W	See section	53.7.13/ 4566
220_00D0	General Purpose Control Register (SATA_GPCR)	32	R/W	0000_0000h	53.7.14/ 4567
220_00D4	General Purpose Status Register (SATA_GPSR)	32	R/W	0000_0000h	53.7.15/ 4568
220_00E0	Timer 1-ms Register (SATA_TIMER1MS)	32	R/W	0001_86A0h	53.7.16/ 4568
220_00F4	Test Register (SATA_TESTR)	32	R/W	0000_0000h	53.7.17/ 4569
220_00F8	Version Register (SATA_VERSIONR)	32	R	3330_302Ah	53.7.18/ 4571
220_0100	Port0 Command List Base Address Register (SATA_P0CLB)	32	R/W	0000_0000h	53.7.19/ 4571

Table continues on the next page...

SATA memory map (continued)

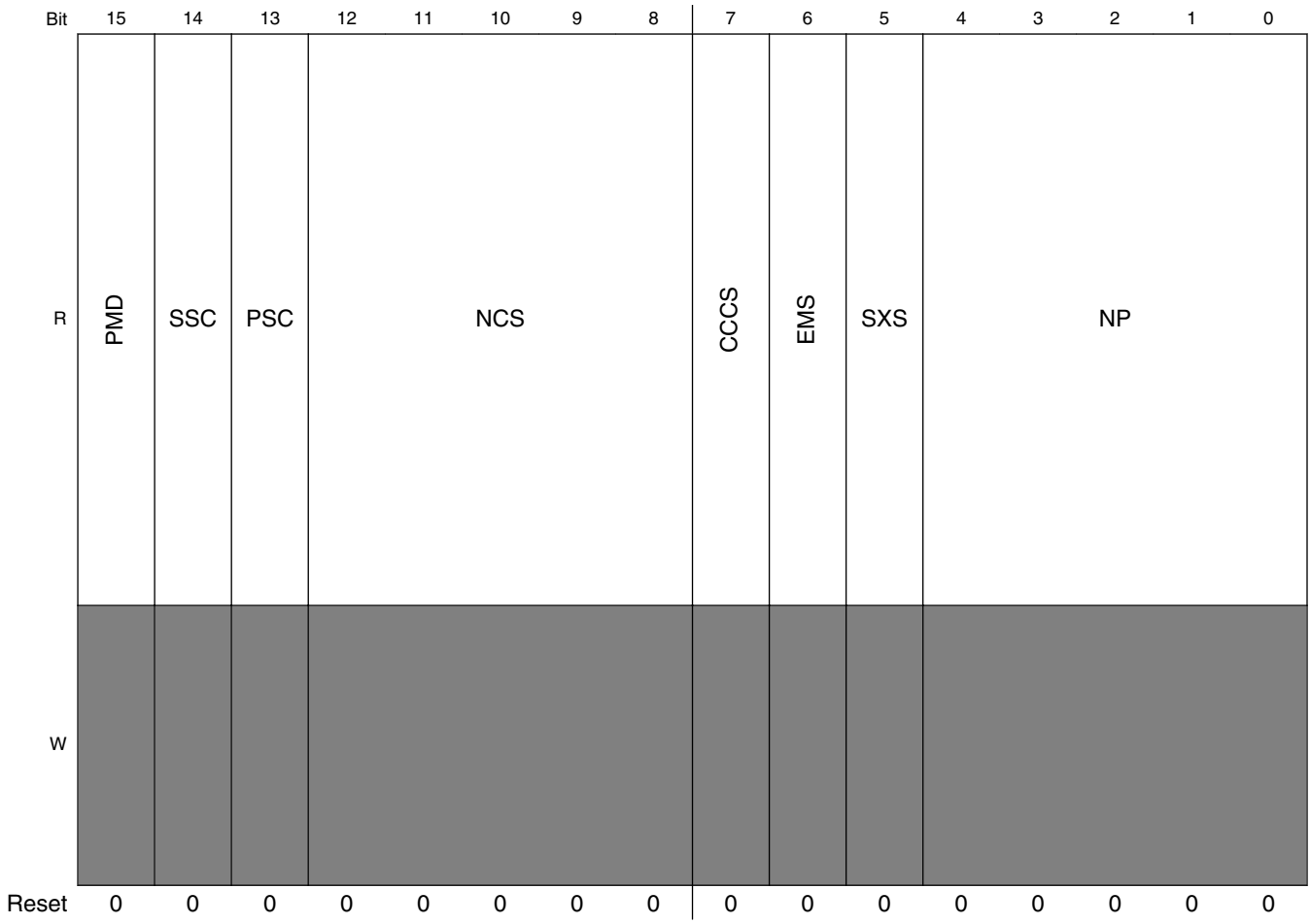
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
220_0108	Port0 FIS Base Address Register (SATA_P0FB)	32	R/W	0000_0000h	53.7.20/ 4572
220_0110	Port0 Interrupt Status Register (SATA_P0IS)	32	R/W	0000_0000h	53.7.21/ 4573
220_0114	Port0 Interrupt Enable Register (SATA_P0IE)	32	R/W	0000_0000h	53.7.22/ 4577
220_0118	Port0 Command Register (SATA_P0CMD)	32	R/W	0000_0000h	53.7.23/ 4580
220_0120	Port0 Task File Data Register (SATA_P0TFD)	32	R	0000_007Fh	53.7.24/ 4584
220_0124	Port0 Signature Register (SATA_P0SIG)	32	R	FFFF_FFFFh	53.7.25/ 4584
220_0128	Port0 Serial ATA Status Register (SATA_P0SSTS)	32	R	0000_0000h	53.7.26/ 4585
220_012C	Port0 Serial ATA Control {SControl} Register (SATA_P0SCTL)	32	R/W	0000_0000h	53.7.27/ 4586
220_0130	Port0 Serial ATA Error Register (SATA_P0SERR)	32	R/W	0000_0000h	53.7.28/ 4588
220_0134	Port0 Serial ATA Active Register (SATA_P0SACT)	32	R/W	0000_0000h	53.7.29/ 4590
220_0138	Port0 Command Issue Register (SATA_P0CI)	32	R/W	0000_0000h	53.7.30/ 4591
220_013C	Port0 Serial ATA Notification Register (SATA_P0SNTF)	32	w1c	0000_0000h	53.7.31/ 4592
220_0170	Port0 DMA Control Register (SATA_P0DMACR)	32	R/W	0000_0044h	53.7.32/ 4592
220_0178	Port0 PHY Control Register (SATA_P0PHYCR)	32	R/W	0000_0000h	53.7.33/ 4594
220_017C	Port0 PHY Status Register (SATA_P0PHYSR)	32	R	0000_0000h	53.7.34/ 4595

53.7.1 HBA Capabilites Register (SATA_CAP)

This register indicates basic capabilities of the SATA block to the software.

Address: 220_0000h base + 0h offset = 220_0000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	S64A	SNCQ	SSNTF	SMPS	SSS	SALP	SAL	SCLO	Reserved				Reserved	SAM	SMP	Reserved
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



SATA_CAP field descriptions

Field	Description
31 S64A	Supports 64-bit Addressing SATA block supports 64-bit addressable data structures by utilizing PFFBU and P#CLBU registers. Reset Value: Configurable. 1 when M_HADDR_WIDTH=64 0 when M_HADDR_WIDTH=32
30 SNCQ	Supports Native Command Queuing. SATA block supports SATA native command queueing by handling DMA Setup FIS natively.
29 SSNTF	Supports SNotification Register. SATA block supports SATA_P 0 SNTF (SNotification) register and its associated functionality.
28 SMPS	Supports Mechanical Presence Switch. This bit is set by the system firmware/BIOS when the platform supports mechanical presence switch for hot plug operation. Dependencies: This field is implemented only when parameter (Macro configuration parm) DEV_MP_SWITCH==Include. When this field is not implemented, this field is reserved, and reads 1'b0.

Table continues on the next page...

SATA_CAP field descriptions (continued)

Field	Description
27 SSS	Supports Staggered Spin-up. This bit is set by the system firmware/BIOS to indicate platform support for staggered devices' spin-up. SATA block supports this feature through the SATA_P 0 CMD[SUD] bit functionality.
26 SALP	Supports Aggressive Link Power Management. SATA block supports auto-generating (Port-initiated) Link Layer requests to the PARTIAL or SLUMBER power management states when there are no commands to process.
25 SAL	Supports Activity LED. SATA block supports activity indication using signal p 0 _act_led.
24 SCLO	Supports Command List Override. SATA block supports the SATA_P 0 CMD[CLO] bit functionality for Port Multiplier devices' enumeration.
23–20 ISS	This field is reserved. Interface Speed Support. Reserved. Returns 0x2 on read.
19 -	This field is reserved. Reserved. Returns 0 on read.
18 SAM	Supports AHCI Mode Only. SATA block supports AHCI mode only and does not support legacy, task-file based register interface.
17 SMP	Supports Port Multiplier. SATA block supports command-based switching Port Multiplier on any of its Ports.
16 -	This field is reserved. Reserved.
15 PMD	PIO Multiple DRQ Block. SATA block supports multiple DRQ block data transfers for the PIO command protocol.
14 SSC	Slumber State Capable. SATA block supports transitions to the interface SLUMBER power management state.
13 PSC	Partial State Capable. SATA block supports transitions to the interface PARTIAL power management state.
12–8 NCS	Number of Command Slots. SATA block supports 32 command slots per Port.
7 CCCS	Command Completion Coalescing Support. SATA block supports command completion coalescing.
6 EMS	Enclosure Management Support. SATA block does not support enclosure management.
5 SXS	Supports External SATA. The options for this field are: 1 Indicates that the SATA block has one or more Ports that has a signal only connector (power is not part of that connector) that is externally accessible. When this bit is set to 1, the software can refer to the SATA_P 0 CMD[ESP] bit to determine whether a specific Port has its signal connector externally accessible. 0 Indicates that the SATA block has no Ports that have a signal only connector externally accessible.

Table continues on the next page...

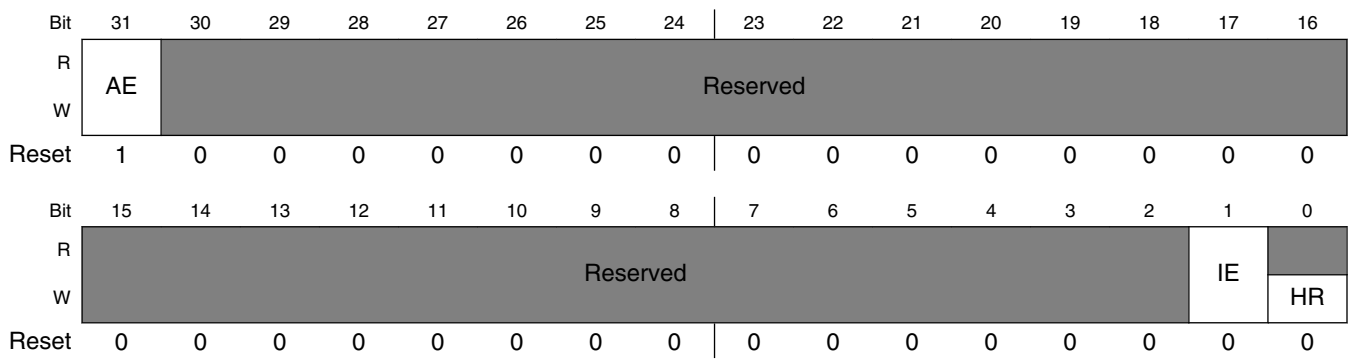
SATA_CAP field descriptions (continued)

Field	Description
	Reset Value: Configurable 1 when any of the SATA_P 0 CMD[ESP]=1 0 when all of the SATA_P 0 CMD[ESP]=0
NP	Number of Ports. 0's based value indicating the number of Ports supported by the SATA block: The options for this field are: • 0x00: 1 Port • 0x01: 2 Ports • 0x02: 3 Ports Reset Value: 0x00

53.7.2 Global HBA Control Register (SATA_GHC)

This register controls various global actions of the SATA block.

Address: 220_0000h base + 4h offset = 220_0004h



SATA_GHC field descriptions

Field	Description
31 AE	AHCI Enable. This bit is always set since SATA block supports only AHCI mode as indicated by the SATA_CAP[SAM]=1.
30–2 -	This field is reserved. Reserved
1 IE	Interrupt Enable. This global bit enables interrupts from the SATA block. When cleared, all interrupt sources from all the Ports are disabled (masked). When set, interrupts are enabled and any SATA block interrupt event causes intrq output assertion.

Table continues on the next page...

SATA_GHC field descriptions (continued)

Field	Description
	This field is reset on Global reset (SATA_GHC[HR]=1).
0 HR	HBA Reset. When set by the software, this bit causes an internal Global reset of the SATA block. All state machines that relate to data transfers and queueing return to an idle state, and all the Ports are re-initialized by sending COMRESET When staggered spin-up is not supported. When staggered spin-up is supported, then the software must spin-up each Port after this reset has completed. See Global Reset for details. The SATA block clears this bit when the reset action is done. A software write of 0 has no effect.

53.7.3 Interrupt Status Register (SATA_IS)

This register indicates which of the Ports within the SATA block have an interrupt pending and require service. This register is reset on Global reset (SATA_GHC[HR]=1).

- Size: 32 bits
- Address offset: 0x08
- Read/write access: Read/Write One to Clear
- Reset: 0x0000_00000

Address: 220_0000h base + 8h offset = 220_0008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved																														IPS	
W																															w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SATA_IS field descriptions

Field	Description
31–2 -	This field is reserved. Reserved.
IPS	Interrupt Pending Status. When bit 1is set, this indicates that Port 0 has an interrupt pending. This bit is set when the Port has an interrupt event pending and the interrupt source is enabled (see the definition of the SATA_P 0 IE register). Bit 0 of the IPS field is not used.

53.7.4 Ports Implemented Register (SATA_PI)

This register indicates which Ports are exposed by the SATA block and are available for the software to use. It is loaded by the BIOS. For example, when the SATA block supports 8 Ports as indicated in the SATA_CAP[NP], only Ports 1, 3, 5, and 7 could be available, while Ports 0, 2, 4, and 6 being unavailable.

NOTE

The contents of this register are relevant to the SATA_CCC_PORTS (Command Completion Coalescing Ports) register.

Address: 220_0000h base + Ch offset = 220_000Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															PI
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- See descriptions for reset values.

SATA_PI field descriptions

Field	Description
31–1 -	This field is reserved. Reserved.
0 PI	Ports Implemented. BIOS must set this bit to 1

53.7.5 AHCI Version Register (SATA_VS)

This register indicates the major and minor version of the AHCI specification that the SATA block implementation supports. The SATA block supports version 1.30.

NOTE

The SATA block core currently complies fully with AHCI version 1.10 , and complies with AHCI version 1.3, except with

respect to FIS-based switching. FIS-based switching is not currently supported.

Address: 220_0000h base + 10h offset = 220_0010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MJR																MNR															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0

SATA_VS field descriptions

Field	Description
31–16 MJR	Major Version Number. Indicates that the major AHCI version is 1.
MNR	Minor Version Number. Indicates that the minor AHCI version is 30.

53.7.6 Command Completion Coalescing Control (SATA_CCC_CTL)

This register is used to configure the command completion coalescing (CCC) feature for the SATA block core. It is reset on Global reset.

Address: 220_0000h base + 14h offset = 220_0014h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	TV															
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CC								INT					Reserved		EN
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- See descriptions for reset values.

SATA_CCC_CTL field descriptions

Field	Description
31–16 TV	Time-out Value. This field specifies the CCC time-out value in 1ms intervals. The software loads this value prior to enabling CCC. The options for this field are:

Table continues on the next page...

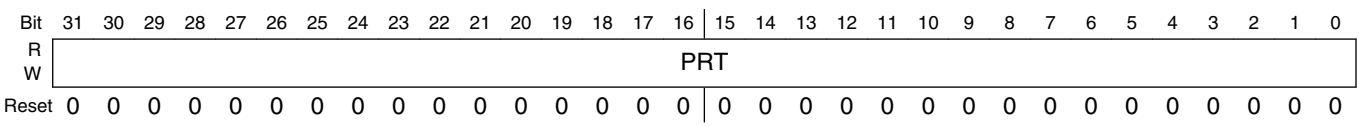
SATA_CCC_CTL field descriptions (continued)

Field	Description
	• RW when SATA_CCC_CTL[EN]==0. • RO when SATA_CCC_CTL[EN]==1. A time-out value of 0x0000 is reserved and should not be used.
15–8 CC	Command Completions. This field specifies the number of command completions that are necessary to cause a CCC interrupt. The value 0x00 for this field disables CCC interrupts being generated based on the number of commands completed. In this case, CCC interrupts are only generated based on the timer. Software loads this value prior to enabling CCC: Field access is: • RW when SATA_CCC_CTL[EN]==0 • RO when SATA_CCC_CTL[EN]==1
7–3 INT	Interrupt. Set this field to 0x01.
2–1 -	This field is reserved. Reserved.
0 EN	Enable. **NOTE:** When field SATA_CCC_CTL[EN]==1, the software can not change the fields SATA_CCC_CTL[TV] and SATA_CCC_CTL[CC]. The options for this field are: 0 CCC feature is disabled and no CCC interrupts are generated. 1 CCC feature is enabled and CCC interrupts may be generated based on the time-out or command completion conditions.

53.7.7 Command Completion Coalescing Ports (SATA_CCC_PORTS)

This register specifies the Ports that are coalesced as part of the command completion coalescing (CCC) feature when SATA_CCC_CTL[EN]==1. It is reset on Global reset.

Address: 220_0000h base + 18h offset = 220_0018h



SATA_CCC_PORTS field descriptions

Field	Description
PRT	Ports. This field is bit significant. Each bit corresponds to a particular Port, where bit 0 corresponds to Port0.

SATA_CCC_PORTS field descriptions (continued)

Field	Description
	Bits set in this register must have the corresponding bit set in the SATA_PI (Ports Implemented Register). The options for this field are: 1 the corresponding Port is part of the CCC feature. 0 the corresponding Port is not part of the CCC feature.

53.7.8 HBA Capabilities Extended Register (SATA_CAP2)

This register indicates capabilites of the SATA block core to the software.

Address: 220_0000h base + 24h offset = 220_0024h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved													APST	Reserved	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

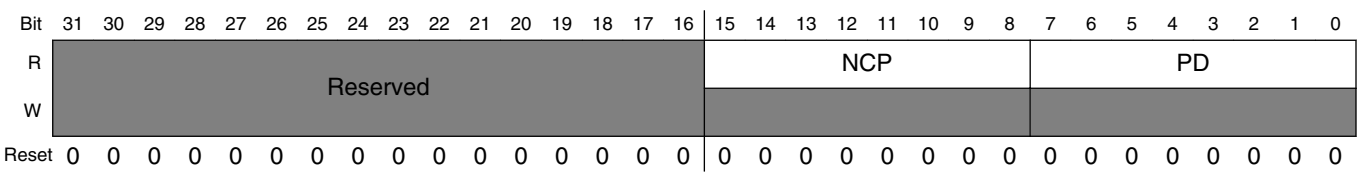
SATA_CAP2 field descriptions

Field	Description
31–3 -	This field is reserved. Reserved
2 APST	Automatic Partial to Slumber Transitions. SATA block supports automatic Partial to Slumber transitions.
-	This field is reserved. Reserved.

53.7.9 BIST Activate FIS Register (SATA_BISTAFR)

This register contains the pattern definition (bits [23:16] of the first DWORD) and data pattern (bits [7:0] of the second DWORD) fields of the received BIST Activate FIS. These fields define the SATA block loopback responder mode requested by the device. It is updated every time a new BIST Activate FIS is received from the device. Reset on Global or Port reset.

Address: 220_0000h base + A0h offset = 220_00A0h



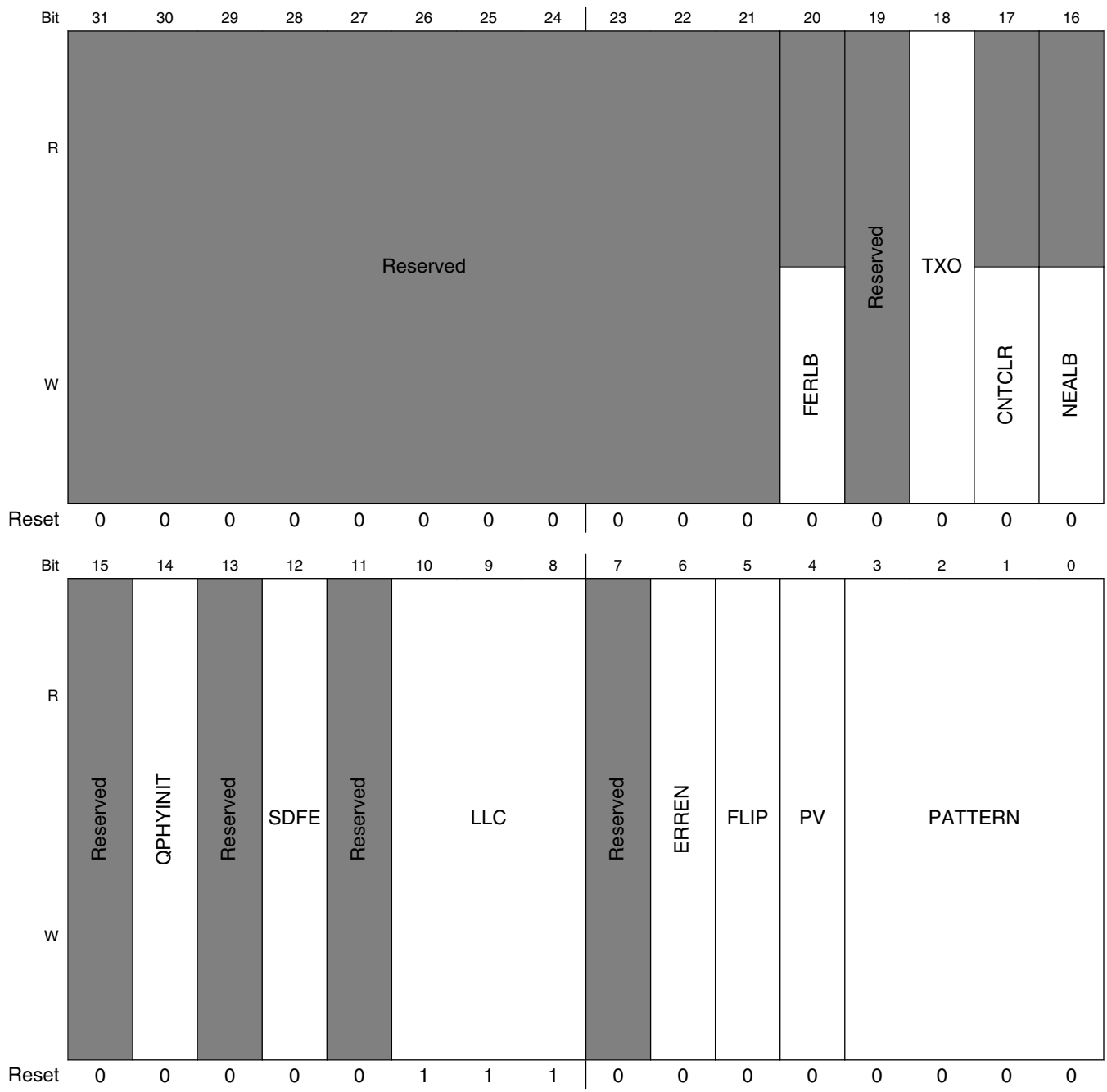
SATA_BISTAFR field descriptions

Field	Description
31–16 -	This field is reserved. Reserved.
15–8 NCP	Least significant byte of the received BIST Activate FIS second DWORD (bits [7:0]). This value defines the required pattern for far-end transmit only mode (SATA_BISTAFR[PD]=0x80 or 0xA0): When none of these values is decoded, the simultaneous switching pattern is transmitted by default. 0xF1 Low transition density pattern (LTDP) 0xB5 High transition density pattern (HTDP) 0xAB Low frequency spectral component pattern (LFSCP) 0x7F Simultaneous switching outputs pattern (SSOP) 0x8B Lone Bit pattern (LBP) 0x78 Mid frequency test pattern (MFTP) 0x4A High frequency test pattern (HFTP) 0x7E Low frequency test pattern (LFTP)
PD	Pattern Definition Indicates the pattern definition field of the received BIST Activate FIS - bits [23:16] of the first DWORD. It is used to put the SATA block in one of the following BIST modes: For far-end transmit only modes SATA_BISTAFR[NCP] field contains the required data pattern. 0x10 Far-end retimed 0x08 Far-end analog (when PHY supports this mode) 0x80 Far-end transmit only 0xA0 Far-end transmit only with scrambler bypassed All other values should not be used by the device, otherwise, the FIS is negatively acknowledged with R_ERRp.

53.7.10 BIST Control Register (SATA_BISTCR)

This register is used in BIST initiator modes. It is loaded by the host software prior to sending BIST Activate FIS to the device (via TXBISTPD write). It is reset on a Global or Port reset.

Address: 220_0000h base + A4h offset = 220_00A4h



SATA_BISTCR field descriptions

Field	Description
31–21 -	This field is reserved. Reserved.
20 FERLB	Far-end Retimed Loopback. When set, this bit is used to put the SATA block Link into Far-end Retimed mode, without the BIST Activate FIS, regardless whether the device is connected or disconnected (Link in NOCOMM state). This field is one-shot type and reads returns 0.
19 -	This field is reserved. Reserved.
18 TXO	Transmit Only. This bit is used to initiate transmission of one of the non-compliant patterns defined by the SATA_BISTCR[PATTERN] value when the device is disconnected.
17 CNTCLR	Counter Clear This bit clears BIST error count registers. This field is one-shot type and reads returns 0. 1 Clear SATA_BISTFCTR, and SATA_BISTSR registers.
16 NEALB	Near-End Analog Loopback This mode should be initiated either in the PARTIAL or SLUMBER power mode, or with the device disconnected from the Port PHY (Link NOCOMM state). BIST Activate FIS is not sent to the device in this mode. This bit places the Port PHY into near-end analog loopback mode. This field is one-shot type and reads returns 0: 1 Near-end analog loopback request. SATA_BISTCR[PATTERN] field contains the appropriate pattern.
15 -	This field is reserved. Reserved.
14 QPHYINIT	When set, this bit enables quick PHY initialization feature. The Link does not require any ALIGNs to transition from OOB to normal operation. **NOTE:** This bit is available only when TX_OOB_MODE = Exclude (0) and ALIGN_MODE = Aligned (1), otherwise it is reserved.
13 -	This field is reserved. Reserved.
12 SDFE	Signal Detect Feature Enable Reset: PHY_INTERFACE_TYPE 1: Link layer feature to handle unstable/absent phy_sig_det signal is enabled 0: Link layer feature to handle unstable/absent phy_sig_det signal is disabled. This bit is set on power-up or asynchronous reset if PHY_INTERFACE_TYPE = Synopsys_SATA_II (1) or PHY_INTERFACE_TYPE = Synopsys_SATA_6G (2), otherwise, the bit is cleared until it is set via programming. It is not affected by a Global reset or COMRESET. **NOTE:** For special handling in systems where phy_sig_det may not be present or stable after OOB signalling and during normal operation . For these systems, phy_rx_data_vld must not be tied high and must go low when no data is detected on the wires.
11 -	This field is reserved. Reserved.
10–8 LLC	Link Layer Control This field controls the Port Link Layer functions: scrambler, descrambler, and repeat primitive drop. Note the different meanings for normal and BIST modes of operation:

Table continues on the next page...

SATA_BISTCR field descriptions (continued)

Field	Description
	• Bit8-SCRAM The options for this field are: 0 Scrambler disabled in normal mode, enabled in BIST mode 1 Scrambler enabled in normal mode, disabled in BIST mode • Bit9-DESCRAM The options for this field are: 0 Descrambler disabled in normal mode, enabled in BIST mode 1 Descrambler enabled in normal mode, disabled in BIST mode • Bit10-RPD The options for this field are: 0 Repeat primitive drop function disabled in normal mode, NA in BIST mode. 1 Repeat primitive drop function enabled in normal mode, NA in BIST mode. The SCRAM bit is cleared (enabled) by the Port when the Port enters a responder far-end transmit BIST mode with scrambling enabled (SATA_BISTAFR[PD]=0x80). In normal mode, the functions scrambler, descrambler, or RPD can be changed only during Port reset (SATA_P 0 SCTL[DET]=0x1)
7 -	This field is reserved. Reserved.
6 ERREN	Error Enable. This bit is used to allow or filter (disable) [internal errors outside the FIS boundary to set corresponding SATA_P 0 SERR bits. The options for this field are: 0 Filter errors outside the FIS, allow errors inside the FIS; 1 Allow errors outside or inside the FIS.
5 FLIP	Flip Disparity This bit is used to change disparity of the current test pattern to the opposite every time its state is changed by the software.
4 PV	Pattern Version This bit is used to select either short or long version of the SSOP, HTDP, LTDP, LFSCP, COMP patterns. The options for this field are: 0 Short pattern version 1 Long pattern version
PATTERN	This field defines one of the following SATA compliant patterns for far-end retimed/ far-end analog/ near-end analog initiator modes, or non-compliant patterns for transmit-only responder mode when initiated by the software writing to the SATA_BISTCR[TXO] bit. If the value is none of the listed below, Composite pattern (COMP) is transmitted by default. 0000b Simultaneous switching outputs pattern (SSOP)

Table continues on the next page...

SATA_BISTCR field descriptions (continued)

Field	Description	
	0001b	High transition density pattern (HTDP)
	0010b	Low transition density pattern (LTDP)
	0011b	Low frequency spectral component pattern (LFSCP)
	0100b	Composite pattern (COMP)
	0101b	Lone bit pattern (LBP)
	0110b	Mid frequency test pattern (MFTP)
	0111b	High frequency test pattern (HFTP)
	1000b	Low frequency test pattern (LFTP)
	All other values	Reserved and should not be used.

53.7.11 BIST FIS Count Register (SATA_BISTFCTR)

This register contains the received BIST FIS count in the loopback initiator far-end retimed, far-end analog and near-end analog modes. It is updated each time a new BIST FIS is received. It is reset by Global reset, Port reset (COMRESET) or by setting the SATA_BISTCR[CNTCLR] bit. This register does not roll over and freezes when the FFFF_FFFFh value is reached. It takes approximately 65 hours of continuous BIST operation to reach this value.

Address: 220_0000h base + A8h offset = 220_00A8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

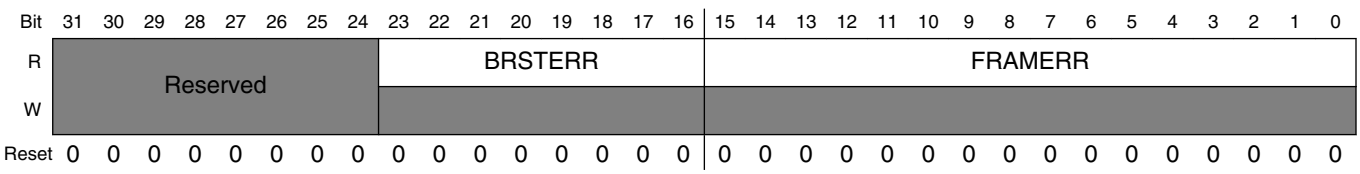
SATA_BISTFCTR field descriptions

Field	Description
-	Received BIST FIS Count

53.7.12 BIST Status Register (SATA_BISTSR)

This register contains errors detected in the received BIST FIS in the loopback initiator far-end retimed, far- end analog and near-end analog modes. It is updated each time a new BIST FIS is received. It is reset by Global reset, Port reset (COMRESET) or by setting the SATA_BISTCR[CNTCLR] bit.

Address: 220_0000h base + ACh offset = 220_00ACh



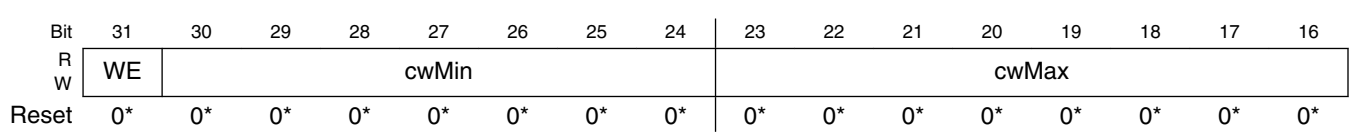
SATA_BISTSR field descriptions

Field	Description
31–24 -	This field is reserved. Reserved.
23–16 BRSTERR	Burst Error. This field contains the burst error count. It is accumulated each time a burst error condition is detected: DWORD error is detected in the received frame and 1.5 seconds (27,000 frames) passed since the previous burst error was detected. The BRSTERR value does not roll over and freezes at FFh. This field is updated when parameter BIST_MODE=DWORD.
FRAMERR	Frame Error. This field contains the frame error count. It is accumulated (new value is added to the old value) each time a new BIST frame with a CRC error is received. The FRAMERR value does not roll over and freezes at FFFFh.

53.7.13 OOB Register (SATA_OOBR)

This register controls the Link layer OOB detection counters. The default values, MIN_COMWAKE, MAX_COMWAKE, MIN_COMINIT and MAX_COMINIT are calculated based on the RXOOB_CLK parameter and loaded on power-up or asynchronous SATA block reset.

Address: 220_0000h base + BCh offset = 220_00BCh



Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ciMin								ciMax							
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- See descriptions for reset values.

SATA_OOBR field descriptions

Field	Description
31 WE	Write Enable This bit is cleared when COMRESET is detected. The options for this field are: 1 SATA_OOBR bits [30:0] can be written 0 SATA_OOBR bits [30:0] are read-only
30–24 cwMin	COMWAKE Minimum Value This field is RW when WE=1 and RO when WE=0.
23–16 cwMax	COMWAKE Maximum Value This field is RW when WE=1 and RO when WE=0.
15–8 ciMin	COMINIT Minimum Value This field is RW when WE=1 and RO when WE=0.
ciMax	COMINIT Maximum Value This field is RW when WE=1 and RO when WE=0.

53.7.14 General Purpose Control Register (SATA_GPCR)

This 32-bit register is used for general purpose control. This register only exists when GP_CTRL parameter is set to “Include” otherwise this location is reserved.

The bits of this register are connected to the corresponding bits of the gp_ctrl output. Resets on power-up (system reset) only to the GP_CTRL_DEF value.

Address: 220_0000h base + D0h offset = 220_00D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GP_CONTROL																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SATA_GPCR field descriptions

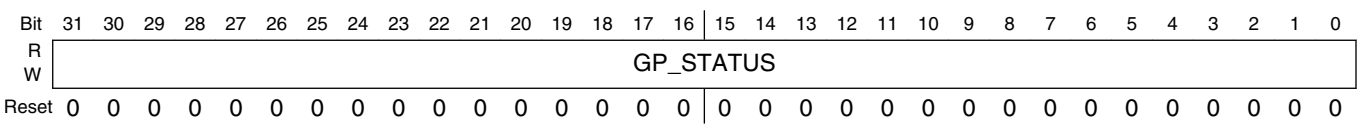
Field	Description
GP_CONTROL	General Purpose Control. Present only when GP_CTRL=Include(1). Reset Value: Configurable parameter GP_CTRL_DEF

53.7.15 General Purpose Status Register (SATA_GPSR)

This 32-bit register is used to monitor the general purpose status. This register only exists when GP_STAT parameter is set to “Include”, otherwise, this location is reserved.

The bits of this register reflect the state of the corresponding bits of the gp_status input. Signals connected to the gp_status input can be asynchronous to any of the DWC_ahsata clocks, however they must not change faster than five hclk/aclk periods, otherwise the GPSR register may never be updated with the intermediate changing values.

Address: 220_0000h base + D4h offset = 220_00D4h



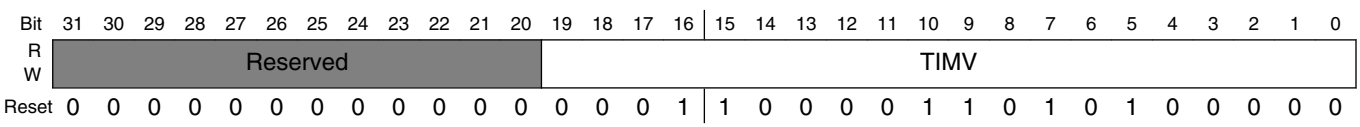
SATA_GPSR field descriptions

Field	Description
GP_STATUS	General Purpose Status. Present only when GP_STAT=Include(1)

53.7.16 Timer 1-ms Register (SATA_TIMER1MS)

This register is used to generate a 1-ms tick for the command completion coalescing (CCC) logic, based on the AHB bus clock frequency. The Software must initialize this register with the required value after power up before using the CCC feature. This register is reset to 100,000 (TIMV value for 100-MHz hclk) on power up and is not affected by Global reset.

Address: 220_0000h base + E0h offset = 220_00E0h



SATA_TIMER1MS field descriptions

Field	Description
31–20 -	This field is reserved. Reserved.
TIMV	1ms Timer Value This field contains the following value for the internal timer to generate 1-ms tick:

Table continues on the next page...

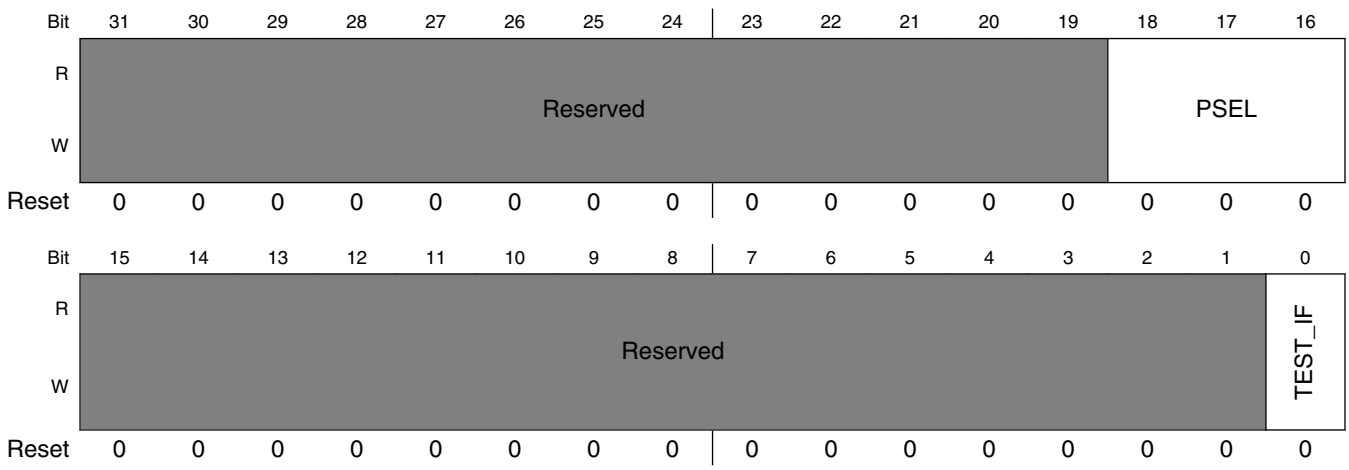
SATA_TIMER1MS field descriptions (continued)

Field	Description
	Fhclk*1000 where Fhclk = AHB clock frequency in MHz The options for this field are: • RW when SATA_CCC_CTL[EN]==0 • RO when SATA_CCC_CTL[EN]==1.

53.7.17 Test Register (SATA_TESTR)

This register is used to put the SATA block slave interface into a test mode and to select a Port for BIST operation.

Address: 220_0000h base + F4h offset = 220_00F4h



SATA_TESTR field descriptions

Field	Description
31–19 -	This field is reserved. Reserved
18–16 PSEL	Port Select This field is used to select a Port for BIST operation: The options for this field are: 0x0 Port0 is selected 0x0 Port1 is selected 0x0 Port2 is selected 0x0 Port3 is selected 0x0 Port4 is selected 0x0 Port5 is selected

Table continues on the next page...

SATA_TESTR field descriptions (continued)

Field	Description
	0x0 Port6 is selected 0x0 Port7 is selected
15–1 -	This field is reserved. Reserved
0 TEST_IF	TEST_IF: Test Interface Normal operation is disabled. The following registers can be accessed in this mode: - SATA_GHC register IE bit - SATA_BISTAFR register NCP and PD bits become read-write - SATA_BISTCR register LLC, ERREN, FLIP, PV, PATTERN - SATA_BISTFCTR, SATA_BISTSR become read-write - SATA_P 0 CLB , SATA_P 0 FB registers - SATA_P 0 IS register RW1C and UFS bits become read-write - SATA_P 0 IE register - SATA_P 0 CMD register ASP, ALPE, DLAE, ATAPI, PMA bits - SATA_P 0 TFD, SATA_P 0 SIG registers become read-write - SATA_P 0 SCTL register - SATA_P 0 SERR register RW1C bits become read-write bits - SATA_P 0 SACT, SATA_P 0 CI, SATA_P 0 SNTF registers become read-write - SATA_P 0 DMACR register - SATA_P 0 PHYCR register - SATA_P 0 PHYSR register becomes read-write Notes: • Interrupt is asserted when any of the SATA_IS register bits is set after setting the corresponding SATA_P 0 IS and SATA_P 0 IE registers and SATA_GHC[IE]=1. • SATA_CAP[SMPS], SATA_CAP[SSS], SATA_PI, SATA_P 0 CMD[ESP], SATA_P 0 CMD[CPD], SATA_P 0 CMD[MPSP], and SATA_P 0 CMD[HPCP] register bits are HwInit type and can not be used in Test mode. They are written once after power-on reset and become read-only. • Global SATA block reset must be issued (SATA_GHC[HR]=1) after TEST_WHEN bit is cleared following the Test mode operation. This bit is used to put the SATA block slave interface into the test mode: The options for this field are: 0 Normal mode: the read back value of some registers is a function of the SATA block state and does not match the value written. 1 Test mode: the read back value of the registers matches the value written.

53.7.18 Version Register (SATA_VERSIONR)

This 32-bit read-only register contains a hard-coded ASCII string that represents the version level of the SATA block. This register contains the ASCII string "300*" (hexadecimal 0x3330302A).

Address: 220_0000h base + F8h offset = 220_00F8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-																															
W																																
Reset	0	0	1	1	0	0	1	1	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	1	0	1	0

SATA_VERSIONR field descriptions

Field	Description
-	SATA block hard-coded hexadecimal version value encoded in ASCII.

53.7.19 Port0 Command List Base Address Register (SATA_P0CLB)

Address: 220_0000h base + 100h offset = 220_0100h

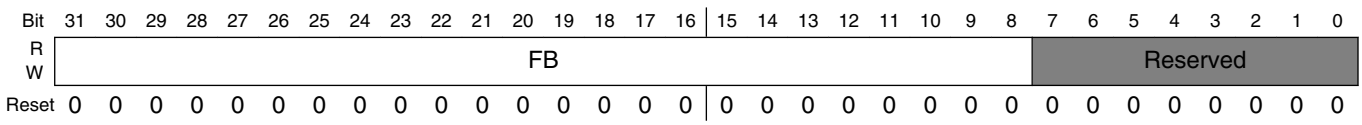
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CLB																						Reserved									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SATA_P0CLB field descriptions

Field	Description
31–10 CLB	Command List Base Address Indicates the 32-bit base physical address for the command list for this Port. This base is used when fetching commands to execute. The structure pointed to by this address range is 1 KB in length. This address must be 1-KB-aligned as indicated by bits [9:0] being read only.
-	This field is reserved. Reserved.

53.7.20 Port0 FIS Base Address Register (SATA_P0FB)

Address: 220_0000h base + 108h offset = 220_0108h



SATA_P0FB field descriptions

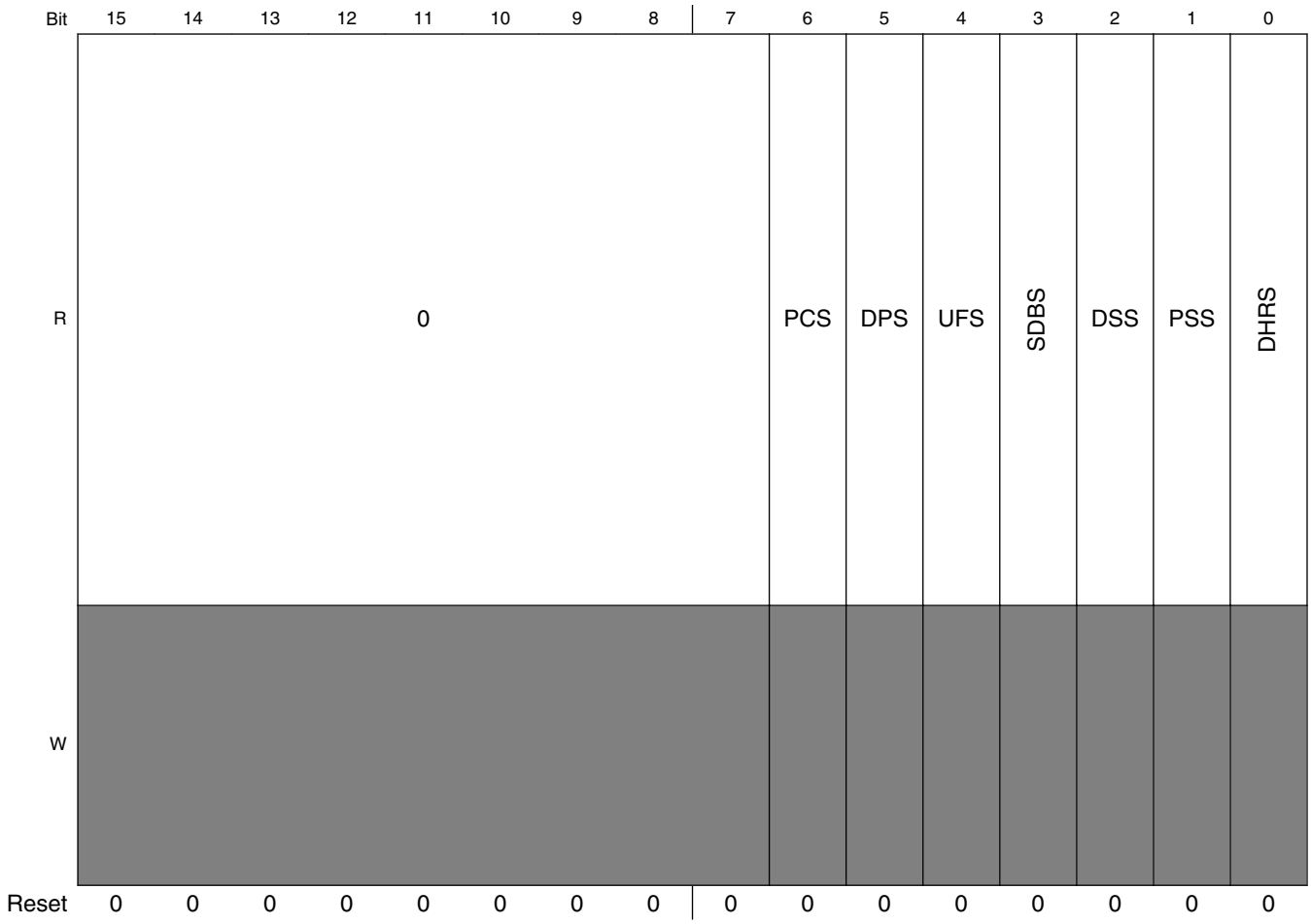
Field	Description
31–8 FB	FIS Base Address. Indicates the 32-bit base physical address for received FISes. The structure pointed to by this address range is 256 bytes in length. This address must be 256byte-aligned as indicated by bits [7:0] being read only. Reset: 0x000000
-	This field is reserved. Reserved.

53.7.21 Port0 Interrupt Status Register (SATA_P0IS)

This register is used to generate SATA block interrupt when any of the bits are set. Bits in this register are set by some internal conditions, and cleared by the software writing ones in the positions it wants to clear. This register is reset on Global SATA block reset.

Address: 220_0000h base + 110h offset = 220_0110h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	TFES	HBFS	HBDS	IFS	INFS	Reserved	OFS	IPMS	PRCS	0					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



SATA_P0IS field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30 TFES	Task File Error Status. This bit is set whenever the SATA_P 0 TFD[STS] register is updated by the device and the error bit (bit 0) is set.
29 HBFS	Host Bus Fatal Error Status. This bit is set when SATA block AHB Master detects an ERROR response from the slave.
28 HBDS	Host Bus Data Error Status. This bit is always cleared to 0.
27 IFS	Interface Fatal Error Status This bit is set when any of the following conditions is detected: • SYNC escape is received from the device during H2D Register or Data FIS transmission; • One or more of the following errors are detected during Data FIS transfer: - 10B to 8B Decode Error (SATA_P 0 SERR[DIAG_B])

Table continues on the next page...

SATA_P0IS field descriptions (continued)

Field	Description
	- Protocol (SATA_P 0 SERR[ERR_P]) - CRC (SATA_P 0 SERR[DIAG_C]) - Handshake (SATA_P 0 SERR[DIAG_H]) - PHY Not Ready (SATA_P 0 SERR[ERR_C]) • Unknown FIS is received with good CRC, but the length exceeds 64 bytes; • PRD table byte count is zero. Port DMA transitions to a fatal state until the software clears SATA_P 0 CMD[ST] bit or resets the interface by way of Port or Global reset.
26 INFS	Interface Non-fatal Error Status This bit is set when any of the following conditions is detected: • One or more of the following errors are detected during non-data FIS transfer - 10B to 8B Decode Error (SATA_P 0 SERR[DIAG_B]) - Protocol (SATA_P 0 SERR[ERR_P]) - CRC (SATA_P 0 SERR[DIAG_C]), - Handshake (SATA_P 0 SERR[DIAG_H]) - PHY Not Ready (SATA_P 0 SERR[ERR_C]); • Command list underflow during read operation (i.e. DMA read) when the software builds command table that has more total bytes than the transaction given to the device.
25 -	This field is reserved. Reserved
24 OFS	Overflow Status This bit is set when command list overflow is detected during read or write operation when the software builds command table that has fever total bytes than the transaction given to the device. Port DMA transitions to a fatal state until the software clears SATA_P 0 CMD[ST] bit or resets the interface by way of Port or Global reset.
23 IPMS	Incorrect Port Multiplier Status. Indicates that the HBA received a FIS from a device whose Port Multiplier field did not match what was expected. This bit may be set during enumeration of devices on a Port Multiplier due to the normal Port Multiplier enumeration process. The software must use the IPMS bit only after enumeration is complete on the Port Multiplier.
22 PRCS	PHY Ready Change Status This bit reflects the state of the SATA_P 0 SERR[DIAG_N] bit. When set to 1, indicates the internal p 0 _phy_ready signal changed state. To clear this bit, the software must clear the SATA_P 0 SERR[DIAG_N] bit to 0.
21–7 Reserved	This read-only field is reserved and always has the value 0.
6 PCS	Port Connect Change Status

Table continues on the next page...

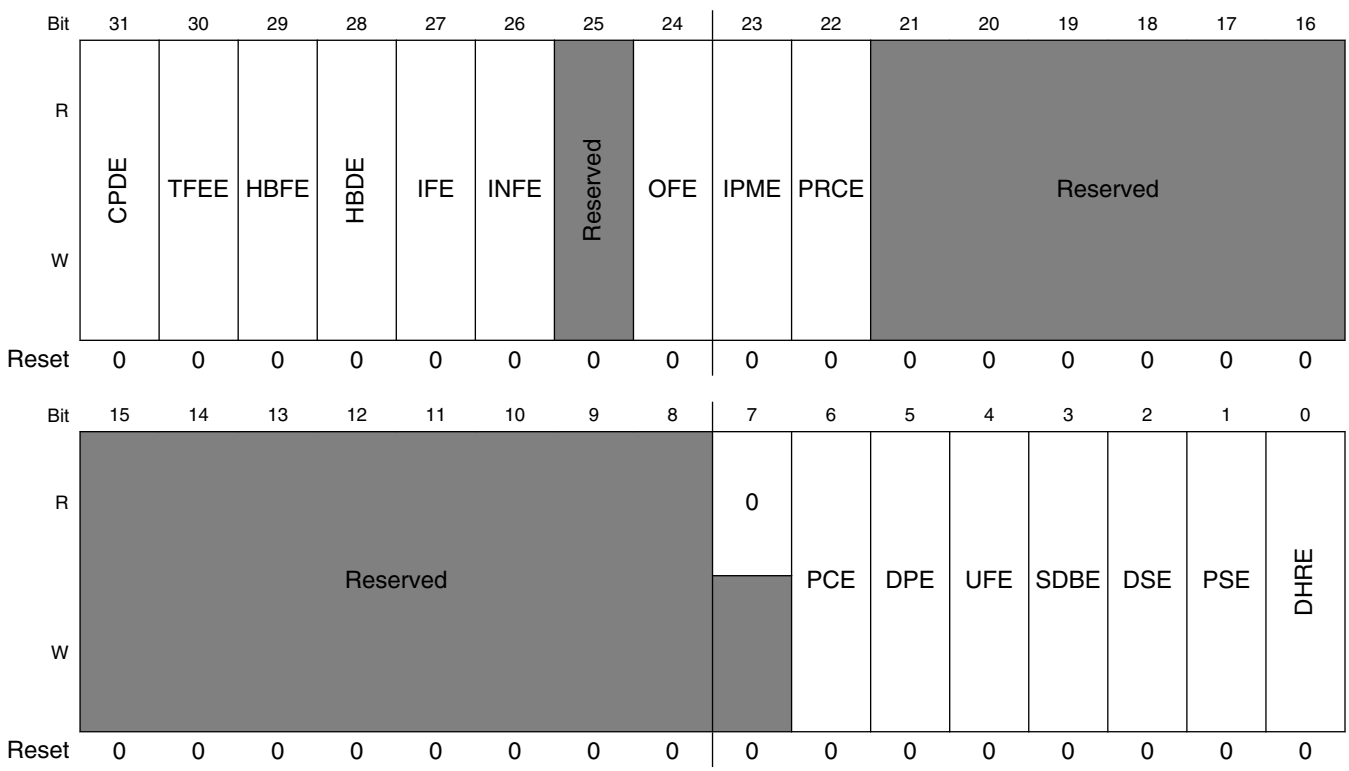
SATA_P0IS field descriptions (continued)

Field	Description
	This bit is cleared only when SATA_P 0 SERR[DIAG_X] is cleared. This bit reflects the state of the SATA_P 0 SERR[DIAG_X] bit: 1 Change in Current Connect Status; 0 No change in Current Connect Status.
5 DPS	Descriptor Processed A PRD with the I bit set has transferred all of its data. **NOTE:** This is an opportunistic interrupt and must not be used to definitively indicate the end of a transfer. Two PRD interrupts could happen close in time together such that the second interrupt is missed when the first PRD interrupt is being cleared.
4 UFS	Unknown FIS Interrupt. When set to 1, indicates that an unknown FIS was received and has been copied into system memory. This bit is cleared to 0 by the software clearing the SATA_P 0 SERR[DIAG_F] bit to 0. **NOTE:** The UFS bit does not directly reflect the SATA_P 0 SERR[DIAG_F] bit. SATA_P 0 SERR[DIAG_F] bit is set immediately when an unknown FIS is detected, whereas the UFS bit is set when that FIS is posted to memory. The software should wait to act on an unknown FIS until the UFS bit is set to 1 or the two bits may become out of sync.
3 SDBS	Set Device Bits Interrupt. A Set Device Bits FIS has been received with the 'I' bit set and has been copied into system memory.
2 DSS	DMA Setup FIS Interrupt A DMA Setup FIS has been received with the 'I' bit set and has been copied into system memory.
1 PSS	PIO Setup FIS Interrupt. A PIO Setup FIS has been received with the 'I' bit set, it has been copied into system memory, and the data related to that FIS has been transferred. **NOTE:** This bit is set even when the data transfer resulted in an error.
0 DHRS	Device to Host Register FIS Interrupt A D2H Register FIS has been received with the 'I' bit set, and has been copied into system memory.

53.7.22 Port0 Interrupt Enable Register (SATA_P0IE)

This register enables and disables the reporting of the corresponding interrupt to the software. When a bit is set (1), and the corresponding interrupt condition is active, then the SATA block intrq output is asserted. Interrupt sources that are disabled (0) are still reflected in the status registers. This register is symmetrical with the SATA_P 0 IS register. This register is reset on Global SATA block reset.

Address: 220_0000h base + 114h offset = 220_0114h



SATA_P0IE field descriptions

Field	Description
31 CPDE	Cold Port Detect Enable Read-only. Returns 0.
30 TFEE	Task File Error Enable Dependencies: when the following conditions are true, the intrq output signal is asserted: • This bit=1 • SATA_GHC[IE]=1 • SATA_P 0 IS[TFES]=1

Table continues on the next page...

SATA_P0IE field descriptions (continued)

Field	Description
29 HBFE	Host Bus Fatal Error Enable Dependencies: when the following conditions are true, the intrq output signal is asserted: • This bit=1 • SATA_GHC[IE]=1 • SATA_P 0 IS[HBFS]=1
28 HBDE	Host Bus Data Error Enable Dependencies: when the following conditions are true, the intrq output signal is asserted: • This bit=1 • SATA_GHC[IE]=1 • SATA_P 0 IS[HBDS]=1
27 IFE	Dependencies: when the following conditions are true, the intrq output signal is asserted: • This bit=1 • SATA_GHC[IE]=1 • SATA_P 0 IS[IFS]=1
26 INFE	Interface Non-Fatal Error Enable Dependencies: when the following conditions are true, the intrq output signal is asserted: • This bit=1 • SATA_GHC[IE]=1 • SATA_P 0 IS[INFS]=1
25 -	This field is reserved. Reserved
24 OFE	Overflow Enable Dependencies: when the following conditions are true, the intrq output signal is asserted: • This bit=1 • SATA_GHC[IE]=1 • SATA_P 0 IS[OFS]=1
23 IPME	Incorrect Port Multiplier Enable Dependencies: when the following conditions are true, the intrq output signal is asserted: • This bit=1 • SATA_GHC[IE]=1 • SATA_P 0 IS[IPMS]=1
22 PRCE	PHY Ready Change Enable Dependencies: when the following conditions are true, the intrq output signal is asserted: • This bit=1 • SATA_GHC[IE]=1 • SATA_P 0 IS[PRCS]=1
21–8 -	This field is reserved. Reserved.

Table continues on the next page...

SATA_P0IE field descriptions (continued)

Field	Description
7 Reserved	This read-only field is reserved and always has the value 0.
6 PCE	Port Change Interrupt Enable Dependencies: when the following conditions are true, the intrq output signal is asserted: • This bit=1 • SATA_GHC[IE]=1 • SATA_P 0 IS[PCS]=1
5 DPE	Descriptor Processed Interrupt Enable Dependencies: when the following conditions are true, the intrq output signal is asserted: • This bit=1 • SATA_GHC[IE]=1 • SATA_P 0 IS[DPS]=1
4 UFE	Unknown FIS Interrupt Enable Dependencies: when the following conditions are true, the intrq output signal is asserted: • This bit=1 • SATA_GHC[IE]=1 • SATA_P 0 IS[UFS]=1
3 SDBE	Set Device Bits FIS Interrupt Enable Dependencies: when the following conditions are true, the intrq output signal is asserted: • This bit=1 • SATA_GHC[IE]=1 • SATA_P 0 IS[SDBS]=1
2 DSE	DMA Setup FIS Interrupt Enable Dependencies: when the following conditions are true, the intrq output signal is asserted: • This bit=1 • SATA_GHC[IE]=1 • SATA_P 0 IS[DSS]=1
1 PSE	PIO Setup FIS Interrupt Enable Dependencies: when the following conditions are true, the intrq output signal is asserted: • This bit=1 • SATA_GHC[IE]=1 • SATA_P 0 IS[PSS]=1
0 DHRE	Device to Host Register FIS Interrupt Dependencies: when the following conditions are true, the intrq output signal is asserted: • This bit=1 • SATA_GHC[IE]=1 • SATA_P 0 IS[DHRS]=1

53.7.23 Port0 Command Register (SATA_P0CMD)

This register contains bits controlling various Port functions. All RW bits are reset on Global reset.

Address: 220_0000h base + 118h offset = 220_0118h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ICC				ASP	ALPE	DLAE	ATAPI	APSTE	Reserved	ESP	Reserved	Reserved	HPCP	PMA	CPS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CR	FR	MPSS	CCS					Reserved			FRE	CLO	POD	SUD	ST
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SATA_P0CMD field descriptions

Field	Description
31–28 ICC	Interface Communication Control This field is used to control power management states of the interface. When the Link layer is currently in the L_IDLE state, writes to this field cause the Port to initiate a transition to the interface power management state requested. When the Link layer is not currently in the L_IDLE state, writes to this field have no effect. • 0xF-0x7: Reserved • 0x6: Slumber. This causes the Port to request a transition of the interface to the Slumber state. The SATA device can reject the request and the interface remains in its current state. • 0x5-0x3: Reserved • 0x2: Partial. This causes the Port to request a transition of the interface to the Partial state. The SATA device can reject the request and the interface remains in its current state. • 0x1: Active. This causes the Port to request a transition of the interface into the active state.

Table continues on the next page...

SATA_P0CMD field descriptions (continued)

Field	Description
	• 0x0: No-Op/ Idle. This value indicates to the software that the Port 0 is ready to accept a new interface control command, although the transition to the previously selected state may not yet have occurred. When the software writes a non-reserved value other than No-Op (0x0), the Port performs the action and update this field back to Idle (0x0). When the software writes to this field to change the state to a state the link is already in (i.e., interface is in the active state and a request is made to go to the active state), the Port takes no action and returns this field to Idle. When the interface is in a low power state and the software wants to transition to a different low power state, the software must first bring the link to active and then initiate the transition to the desired low power state.
27 ASP	Aggressive Slumber/ Partial The options for this field are: • When set to 1, and SATA_P 0 CMD[ALPE]=1, the Port aggressively enters the SLUMBER state when one of the following conditions is true: - The Port clears the SATA_P 0 CI and the SATA_P 0 SACT register is cleared. - The Port clears the SATA_P 0 SACT register and SATA_P 0 CI is cleared. • When cleared to 0, and SATA_P 0 CMD[ALPE]=1, the Port aggressively enters the PARTIAL state when one of the following conditions is true: - The Port clears the SATA_P 0 CI register and the SATA_P 0 SACT register is cleared. - The Port clears the SATA_P 0 SACT register and SATA_P 0 CI is cleared.
26 ALPE	Aggressive Link Power Management Enable When set to 1, the Port aggressively enters a lower link power state (PARTIAL or SLUMBER) based on the setting of the SATA_P 0 CMD[ASP] bit. When cleared to 0, aggressive power management state transition is disabled.
25 DLAE	Drive LED on ATAPI Enable When set to 1, SATA_P 0 CMD[ATAPI]=1, and commands are active, the Port asserts p 0 _act_led output.
24 ATAPI	ATAPI Device is ATAPI This bit is used by the Port to control whether to assert p 0 _act_led output when commands are active. The options for this field are: 0 non-ATAPI device 1 ATAPI device
23 APSTE	Device is ATAPI This bit is used by the Port to control whether to assert p 0 _act_led output when commands are active. The options for this field are: 0 non-ATAPI device 1 ATAPI device
22 -	This field is reserved. Reserved
21 ESP	External SATA Port When set to 1, indicates that this Port's signal only connector is externally accessible. When set to 1, SATA_CAP[SXS] is also set to 1.

Table continues on the next page...

SATA_P0CMD field descriptions (continued)

Field	Description
	When cleared to 0, indicates that this Port's signal only connector is not externally accessible. Note: The ESP bit is mutually exclusive with SATA_P 0 CMD[HPCP]
20 -	This field is reserved. Reserved. Returns 0 on read.
19 -	This field is reserved. Reserved. Returns 0 on read.
18 HPCP	Hot Plug Capable Port **NOTE:** The HPCP bit is mutually exclusive with SATA_P 0 CMD[ESP]. The options for this field are: 1 Indicates that this Port's signal and power connectors are externally accessible via a joint signal-power connector for blindmate device hot plug. 0 Indicates that this Port's signal and power connectors are not externally accessible.
17 PMA	Port Multiplier Attached The software is responsible for detecting whether a Port Multiplier is present; the SATA block Port does not auto-detect the presence of a Port Multiplier. The options for this field are: 1 A Port Multiplier is attached to this Port. 0 A Port Multiplier is not attached to this Port.
16 CPS	Cold Presence State This bit reports whether a device is currently detected on this Port as indicated by the p 0 _cp_det input state (assuming SATA_P 0 CMD[CPD]=1). The options for this field are: 1 device is attached to this Port 0 no device attached to this Port
15 CR	Command List Running When this bit is set to '1', the command list DMA engine for this Port is running. See AHCI state machine in AHCI specification section 5.3.2 for details on when this bit is set and cleared by the Port.
14 FR	FIS Receive Running When set to '1', the FIS Receive DMA engine for the Port is running. See AHCI specification section 10.3.2 for details on when this bit is set and cleared by the Port.
13 MPSS	Mechanical Presence Switch State The software must use this bit only when both SATA_CAP[SMPS] and SATA_P 0 CMD[MPSP] are set. This bit reports the state of a mechanical presence switch attached to this Port as indicated by the p 0 _mp_switch input state (assuming SATA_CAP[SMPS]=1 and SATA_P 0 CMD[MPSP]=1). The options for this field are: When SATA_CAP[SMPS]=0 then this bit is cleared to 0. 0 Switch is closed 1 Switch is open
12–8 CCS	Current Command Slot This field is set to the command slot value value of the command that is currently being issued by the Port.

Table continues on the next page...

SATA_P0CMD field descriptions (continued)

Field	Description
	• When SATA_P 0 CMD[ST] transitions from 1 to 0, this field is recleared to 0x00. • After SATA_P 0 CMD[ST] transitions from 0 to 1, the highest priority slot to issue from next is command slot 0. After the first command has been issued, the highest priority slot to issue from next is SATA_P 0 CMD[CCS]+1.For example, after the Port has issued its first command, when CCS=0x00 and SATA_P 0 CI is cleared to 0x3, the next command issued is from command slot 1. This field is valid only when SATA_P 0 CMD[ST] is set to 1.
7–5 -	This field is reserved. Reserved.
4 FRE	FIS Receive Enable When set to 1, the Port may post received FISes into the FIS receive area pointed to by SATA_P 0 FB . When cleared, received FISes are not accepted by the Port, except for the first D2H register FIS after the initialization sequence, and no FISes are posted to the FIS receive area. The software must not set this bit until SATA_P 0 FB has been programmed with a valid pointer to the FIS receive area When the software wishes to move the base, this bit must first be cleared, and the software must wait for the SATA_P 0 CMD[FR] bit to be cleared.
3 CLO	Command List Override Setting this bit to 1 causes the SATA_P 0 TFD[STS] field BSY bit and the SATA_P 0 TFD[STS] field DRQ bit to be cleared. This allows a software reset to be transmitted to the device regardless of whether the BSY and DRQ bits are still set in the SATA_P 0 TFD[STS] field. This bit is cleared to 0 when SATA_P 0 TFD[STS] BSY bit and SATA_P 0 TFD[STS] DRQ bit have been cleared. A write to this register with a value of '0' has no effect. This bit should only be set to 1 immediately prior to setting SATA_P 0 CMD[ST] bit to 1 from a previous value of 0. Setting this bit to 1 at any other time is not supported and results in indeterminate behavior.
2 POD	Power On Device This bit is read/write when cold presence detection is supported on this Port as indicated by SATA_P 0 CMD[CPD]=1. This bit is read-only 1 when cold presence detection is not supported and SATA_P 0 CMD[CPD]=0. When set, the Port asserts the p 0 _cp_pod output pin so that it may be used to provide power to a cold-presence detectable Port.
1 SUD	Spin-Up Device This bit is read/write when staggered spin-up is supported as indicated by the SATA_CAP[SSS]=1. This bit is read-only 1 when staggered spin-up is not supported and SATA_CAP[SSS]=0. On an edge detect from 0 to 1, the Port starts a COMRESET initialization sequence to the device. Clearing this bit causes no action on the interface. Note: The SUD bit is read-only 0 on power-up until SATA_CAP[SSS] bit is written with the required value.
0 ST	Start When set to 1, the Port processes the command list. When cleared, the Port does not process the command list. Whenever this bit is changed from a 0 to a 1, the Port starts processing the command list at entry'0. Whenever this bit is changed from a 1 to a 0, the SATA_P 0 CI register is cleared by the Port upon transition into an idle state. Refer to AHCI specification, section 10.3.1, for important restrictions on when this bit can be set to 1. Note: SATA_P 0 SERR register must be cleared prior to setting ST bit to 1.

53.7.24 Port0 Task File Data Register (SATA_P0TFD)

This register contains Error and Status registers updated every time a new Register FIS, PIO Setup FIS, or Set Device Bits FIS is received from the device. Reset on Global or Port reset (COMRESET).

Address: 220_0000h base + 120h offset = 220_0120h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved																ERR								STS							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1

SATA_P0TFD field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
15–8 ERR	Error This field contains the latest copy of the task file error register.
STS	Status This field contains the latest copy of the task file status register. The bits that affect SATA block operation are: • Bit [7] BSY - Indicates the interface is busy • Bits [6:4] cs - Command specific • Bit [3] DRQ - Indicates a data transfer is requested • Bits [2:1] cs - Command specific • Bit [0] ERR - Indicates an error during the transfer **NOTE:** The Port updates the entire 8-bit field, not just the bits noted above.

53.7.25 Port0 Signature Register (SATA_P0SIG)

Address: 220_0000h base + 124h offset = 220_0124h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SIG																															
W																																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

SATA_P0SIG field descriptions

Field	Description
SIG	Signature This field contains the signature received from a device on the first D2H Register FIS. The bit order as follows: • Bits [31:24] - LBA High (Cylinder High) Register • Bits [23:16] - LBA Mid (Cylinder Low) Register • Bits [15:8] - LBA Low (Sector Number) Register • Bits [7:0] - Sector Count Register This field is updated once after a reset sequence. Reset on Global or Port reset.

53.7.26 Port0 Serial ATA Status Register (SATA_P0SSTS)

This 32-bit register conveys the current state of the interface and host. The Port updates it continuously and asynchronously. When the Port transmits a COMRESET to the device, this register is updated to its reset values (i.e., Global reset, Port reset, or COMINIT from the device

Address: 220_0000h base + 128h offset = 220_0128h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved																				IPM				SPD				Reserved			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SATA_P0SSTS field descriptions

Field	Description
31–12 -	This field is reserved. Reserved
11–8 IPM	Interface Power Management Indicates the current interface state. The options for this field are: 0x0 Device not present or communication not established 0x1 Interface in active state 0x2 Interface in Partial power management state 0x6 Interface in Slumber power management state All other values Reserved
7–4 SPD	Current Interface Speed Indicates the negotiated interface communication speed. The options for this field are: 0x0 Device not present or communication not established

Table continues on the next page...

SATA_P0SSTS field descriptions (continued)

Field	Description
	0x1 1.5 Gb/s communication rate negotiated 0x2 3.0 Gb/s communication rate negotiated All other values Reserved and should not be used
DET	This field is reserved. Indicates the interface device detection and PHY state. The options for this field are: • 0x0: No device detected and PHY communication not established • 0x1: Device presence detected but PHY communication not established (COMINIT is detected) • 0x3: Device presence detected and PHY communication established (“PHY Ready” is detected) • 0x4: PHY in offline mode as a result of the interface being disabled or running in a BIST loopback mode. All other values reserved.

53.7.27 Port0 Serial ATA Control {SControl} Register (SATA_P0SCTL)

This 32-bit read-write register is used by the software to control SATA interface capabilities. Writes to this register result in an action being taken by the Port PHY interface. Reads from the register return the last value written to it. Reset on Global reset.

These bits are static and should not be changed frequently due to the clock crossing between the Transport and Link Layers. The software must wait for at least seven periods of the slower clock (clk_asic 0 or hclk) before changing this register

Address: 220_0000h base + 12Ch offset = 220_012Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																				IPM				SPD				DET			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SATA_P0SCTL field descriptions

Field	Description
31–12 -	This field is reserved. Reserved
11–8 IPM	Interface Power Management Transitions Allowed This field indicates which power states the Port PHY interface is allowed to transition to. When an interface power management state is disabled, the Port does not initiate that state and any request from the device to enter that state is rejected via PMNAKp The options for this field are: 0x0 No interface power management state restrictions 0x1 Transitions to the Partial state disabled 0x2 Transitions to the Slumber state disabled

Table continues on the next page...

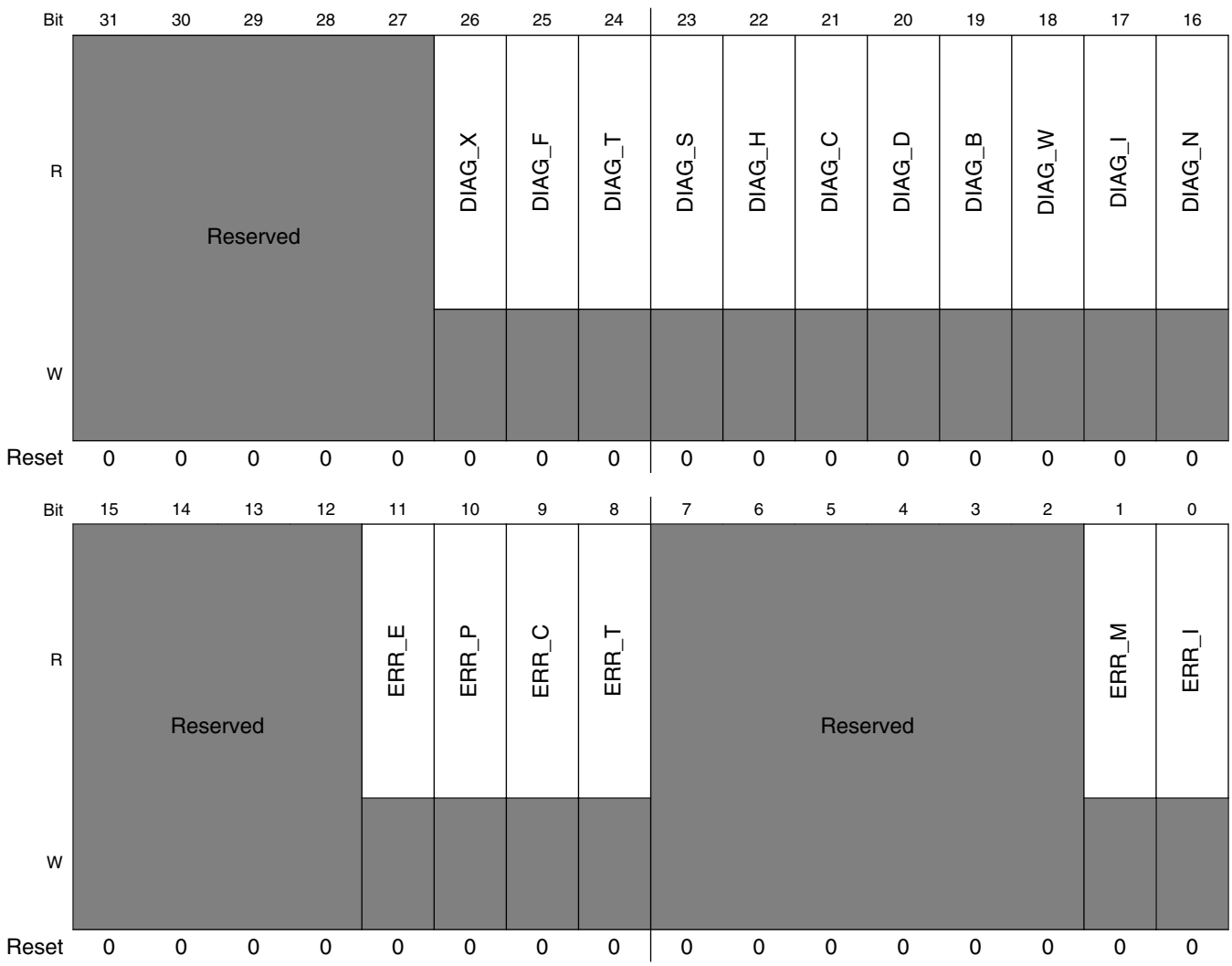
SATA_P0SCTL field descriptions (continued)

Field	Description
	0x3 Transitions to both Partial and Slumber states disabled All other values. Reserved and should not be used
7–4 SPD	Speed Allowed This field indicates the highest allowable speed of the Port PHY interface. The options for this field are: **NOTE:** When the host software must change this field value, the host must also reset the Port (SATA_P 0 SCTL[DET] = 0x1) at the same time to ensure proper speed negotiation. 0x0 No speed negotiation restrictions 0x1 Limit speed negotiation to SATA 1.5 Gb/s communication rate 0x2 Limit speed negotiation to SATA 3.0 Gb/s communication rate All other values Reserved and should not be used.
DET	Device Detection Initialization Controls the Port's device detection and interface initialization. The options for this field are: **NOTE:** This field may only be modified when SATA_P 0 CMD[ST] is 0. Changing this field while the SATA_P 0 CMD[ST]=1 results in undefined behavior. When SATA_P 0 CMD[ST] is set to 1, this field should have a value of 0x0. 0x0 No device detection or initialization action requested 0x1 Perform interface initialization sequence to establish communication. This results in the interface being reset and communication re initialized. 0x4 Disable the Serial ATA interface and put the Port PHY in offline mode. All other values reserved.

53.7.28 Port0 Serial ATA Error Register (SATA_P0SERR)

This 32-bit register represents all the detected interface errors accumulated since the last time it was cleared. The set bits in the SError register indicate that the corresponding error condition became true one or more times since the last time the bit was cleared. The set bits in this register are explicitly cleared by a write operation to the register, Global reset, or Port reset (COMRESET). The value written to clear the set error bits should have ones encoded in the bit positions corresponding to the bits that are to be cleared. All bits in the following table have a reset value of 0.

Address: 220_0000h base + 130h offset = 220_0130h



SATA_P0SERR field descriptions

Field	Description
31–27 -	This field is reserved. Reserved
26 DIAG_X	Exchanged This bit is set to 1 when PHY COMINIT signal is detected. This bit is reflected in the SATA_P 0 IS[PCS] bit.
25 DIAG_F	Unknown FIS Type This bit indicates that one or more FISes were received by the Transport layer with good CRC, but had a type field that was not recognized/known and the length was less than or equal to 64bytes. **NOTE:** When the Unknown FIS length exceeds 64 bytes, the DIAG_F bit is not set and the DIAG_T bit is set instead.
24 DIAG_T	Transport State Transition Error This bit indicates that a Transport Layer protocol violation was detected since the last time this bit was cleared. See Transport Check (TCHK) for details.
23 DIAG_S	Link Sequence Error This bit indicates that one or more Link state machine error conditions was encountered. One of the conditions that cause this bit to be set is device doing SYNC escape during FIS transmission.
22 DIAG_H	Handshake Error This bit indicates that one or more R_ERRp was received in response to frame transmission. Such errors may be the result of a CRC error detected by the device, a disparity or 8b/10b decoding error, or other error condition leading to a negative handshake on a transmitted frame.
21 DIAG_C	CRC Error
20 DIAG_D	Disparity Error This bit is always cleared to 0 since it is not used by the AHCI specification.
19 DIAG_B	10B to 8B Decode Error This bit indicates errors were detected by 10b8b decoder. This bit indicates that one or more CRC errors were detected by the Link layer during FIS reception. **NOTE:** This bit is set only when an error is detected on the received FIS data dword. This bit is not set when an error is detected on the primitive, regardless whether it is inside or outside the FIS.
18 DIAG_W	Comm Wake This bit is set when PHY COMWAKE signal is detected.
17 DIAG_I	PHY Internal Error This bit is set when the PHY detects some internal error as indicated by the assertion of the p 0 _phy_rx_err input. **NOTE:** The setting of this bit is controlled by the SATA_BISTCR[ERREN] bit: when ERREN==0 (default), only errors occurring inside the received FIS cause DIAG_I bit to be set; when ERREN==1, any error inside or outside the FIS causes the DIAG_I bit to be set.
16 DIAG_N	PHY Ready Change This bit indicates that the PHY Ready signal changed state. This bit is reflected in the SATA_P 0 IS[PRCS] bit.
15–12 -	This field is reserved. Reserved

Table continues on the next page...

SATA_P0SERR field descriptions (continued)

Field	Description
11 ERR_E	Internal Error This bit is set to 1 when one or more AHB bus ERROR responses are detected on the master interface.
10 ERR_P	Protocol Error This bit is set to 1 when any of the following conditions are detected. • Transport state transition error (DIAG_T) • Link sequence error (DIAG_S) • RxFIFO overflow • Link bad end error (WTRM instead of EOF is received).
9 ERR_C	Non-Recovered Persistent Communication Error This bit is set to 1 when PHY Ready signal is negated due to the loss of communication with the device or problems with interface, but not after transition from active to Partial or Slumber power management state.
8 ERR_T	Non-Recovered Transient Data Integrity Error This bit is set when any of the following SATA_P 0 SERR register bits is set during Data FIS transfer: ERR_P (Protocol) • DIAG_C (CRC) • DIAG_H (Handshake) • ERR_C ("PHY Ready" negation)
7–2 -	This field is reserved. Reserved
1 ERR_M	Recovered Communication Error This bit is set to 1 when PHY Ready condition is detected after interface initialization, but not after transition from Partial or Slumber power management state to active state.
0 ERR_I	This bit is set when any of the following SATA_P 0 SERR register bits is set during non- Data FIS transfer: • DIAG_C (CRC) • DIAG_H (Handshake) • ERR_C ("PHY Ready" negation)

53.7.29 Port0 Serial ATA Active Register (SATA_P0SACT)

Address: 220_0000h base + 134h offset = 220_0134h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SATA_P0SACT field descriptions

Field	Description
DS	Device Status

SATA_P0SACT field descriptions (continued)

Field	Description
	This field is bit significant. Each bit corresponds to the TAG and command slot of a native queued command, where bit 0 corresponds to TAG 0 and command slot 0. Software sets this field prior to issuing a native queued command for a particular command slot. Prior to writing SATA_P 0 CI[TAG] to 1, the software sets DS[TAG] to 1 to indicate that a command with that TAG is outstanding. This field is cleared to 0 when: • The software writes SATA_P 0 CMD[ST from a 1 to a 0 . • The device sends a Set Device Bits FIS to the Port. The Port clears bits in this field that are set in the SActive field of the Set Device Bits FIS. The Port clears only bits that correspond to native queued commands that have completed successfully. This field is not cleared by the following: • Port reset (COMRESET). • Software reset. **NOTE:** Software must write this field only when SATA_P 0 CMD[ST] bit is set to 1.

53.7.30 Port0 Command Issue Register (SATA_P0CI)

Address: 220_0000h base + 138h offset = 220_0138h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CI																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SATA_P0CI field descriptions

Field	Description
CI	Command Issued This field is bit significant. Each bit corresponds to a command slot, where bit 0 corresponds to command slot 0. This field is set by the software to indicate to the Port that a command has been built in system memory for a command slot and may be sent to the device. When the Port receives a FIS which clears the BSY, DRQ, and ERR bits for the command, it clears the corresponding bit in this register for that command slot. Bits in this field can only be set to 1 by the software when SATA_P 0 CMD[ST] is set to 1. **NOTE:** This field is reset when SATA_P 0 CMD[ST] is written from a 1 to a 0 by the software.

53.7.31 Port0 Serial ATA Notification Register (SATA_P0SNTF)

This register is used to determine when asynchronous notification events have occurred for directly connected devices and devices connected to a Port Multiplier.

Address: 220_0000h base + 13Ch offset = 220_013Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved																PMN															
W																	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SATA_P0SNTF field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
PMN	PM Notify This field indicates whether a particular device with the corresponding PM Port number issued a Set Device Bits FIS to the SATA block Port with the Notification bit set: • PM Port 0h sets bit 0, • PM Port 1h sets bit 1, ... • PM Port Fh sets bit 15. Individual bits are cleared by the software writing 1s to the corresponding bit positions. This field is reset on Global reset, but it is not reset by Port reset (COMRESET) or software reset.

53.7.32 Port0 DMA Control Register (SATA_P0DMACR)

This register contains bits for controlling the Port DMA engine. The software can change the fields of this register only when SATA_P 0 CMD[ST]=0. Power-up (system reset), Global reset, or Port reset (COMRESET) reset this register to the default value.

Address: 220_0000h base + 170h offset = 220_0170h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																-				-				RXTS				TXTS			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0

SATA_P0DMACR field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
15–12 -	Reserved
11–8 -	Reserved
7–4 RXTS	Receive Transaction Size This field defines the Port DMA transaction size in DWORDs for receive (system bus write, device read) operation. This field is read-write when SATA_P 0 CMD[ST]=0 and read-only when SATA_P 0 CMD[ST]=1. The maximum value of this field is determined by the RxFIFO depth parameter. When the software attempts to write a value exceeding this value, the maximum value would be set instead. 0x0 1 DWORD 0x1 2 DWORD 0x2 4 DWORD 0x3 8 DWORD 0x4 16 DWORDs (maximum value when RXFIFO_DEPTH=64) 0x5 32 DWORD 0x6 64 DWORDs (maximum value when RXFIFO_DEPTH=128) 0x7 128 DWORDs (maximum value when RXFIFO_DEPTH=256) 0x8 256 DWORDs (maximum value when RXFIFO_DEPTH=512) 0x9 12 DWORDs (maximum value when RXFIFO_DEPTH=1024) 0xA 1024 DWORDs (maximum value when RXFIFO_DEPTH=2048) All other values are reserved and should not be used.
TXTS	Transmit Transaction Size This field defines the DMA transaction size in DWORDs for transmit (system bus read, device write) operation. The options for this field are: This field is read-write when SATA_P 0 CMD[ST]=0 and read-only when SATA_P 0 CMD[ST]=1. The maximum value of this field is determined by the TxFIFO depth parameter. When the software attempts to write a value exceeding this value, the maximum value would be set instead. 0x0 1 DWORD 0x1 2 DWORD 0x2 4 DWORD 0x3 8 DWORD 0x4 16 DWORDs (maximum value when TXFIFO_DEPTH=32) 0x5 32 DWORDs (maximum value when TXFIFO_DEPTH=64) 0x6 64 DWORDs (maximum value when TXFIFO_DEPTH=128) 0x7 128 DWORDs (maximum value when TXFIFO_DEPTH=256) 0x8 256 DWORDs (maximum value when TXFIFO_DEPTH=512) 0x9 512 DWORDs (maximum value when TXFIFO_DEPTH=1024) 0xA 1024 DWORDs (maximum value when TXFIFO_DEPTH=2048) All other values are reserved and should not be used.

53.7.33 Port0 PHY Control Register (SATA_P0PHYCR)

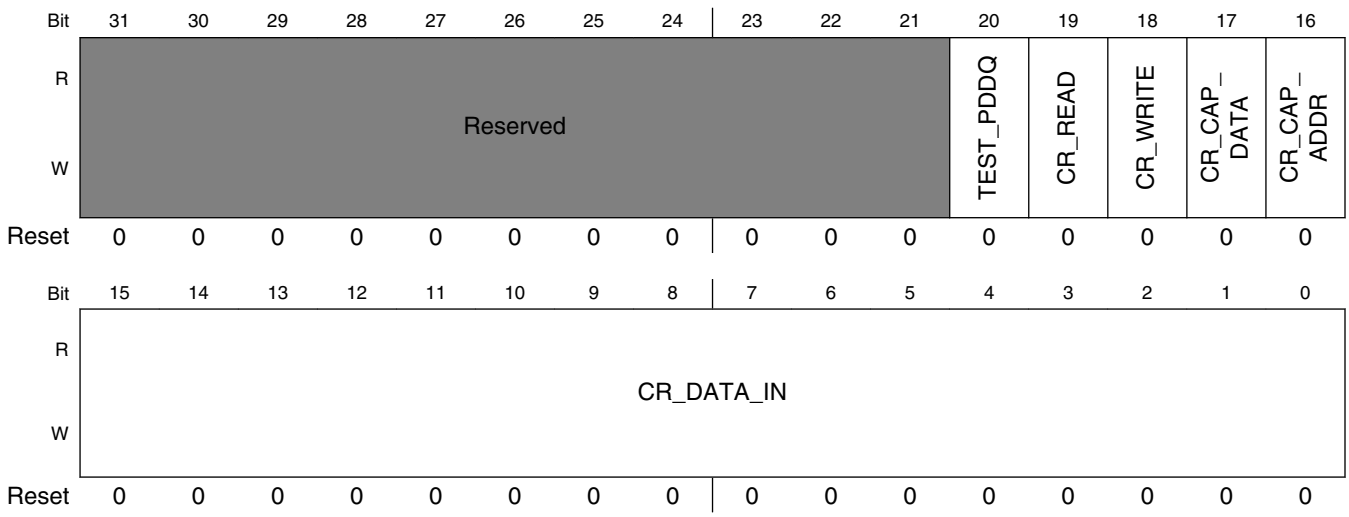
This register is used for Port PHY control.

Bits of this register are connected to the corresponding bits of the p 0 _phy_ctrl output Port.

NOTE

The SATA_P 0 PHYCR register supports only 32-bit write access

Address: 220_0000h base + 178h offset = 220_0178h



SATA_P0PHYCR field descriptions

Field	Description
31–21 -	This field is reserved. Reserved
20 TEST_PDDQ	Test IDDQ
19 CR_READ	CR Read. Reads from the referenced Address register.
18 CR_WRITE	CR Write. Writes the Write Data register to the referenced Address register.
17 CR_CAP_DATA	CR Capture Data. Captures phy_cr_data_in[15:0] into the Write Data register.
16 CR_CAP_ADDR	CR Capture Address. Captures phy_cr_data_in[15:0] into the Address register.
CR_DATA_IN	CR Address and Write Data Input Bus. Supplies and captures address and write data.

53.7.34 Port0 PHY Status Register (SATA_P0PHYSR)

This register is used to monitor PHY status.

The bits of this register reflect the state of the corresponding bits of the p 0 _phy_status input.

Signals connected to the p 0 _phy_status input can be asynchronous to any of the SATA block clocks, however they must not change faster than five hclk periods, otherwise the SATA_P 0 PHYSR register may never be updated with the intermediate changing values.

Address: 220_0000h base + 17Ch offset = 220_017Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0													CR_ ACK	0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CR_DATA_OUT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SATA_P0PHYSR field descriptions

Field	Description
31–19 Reserved	This read-only field is reserved and always has the value 0.
18 CR_ACK	CR Acknowledgement. Acknowledgement for the phy_cr_cap_addr, phy_cr_cap_data, phy_cr_write, and phy_cr_read control signals.
17–16 Reserved	This read-only field is reserved and always has the value 0.
CR_DATA_OUT	CR Data Output Bus. Always presents last read data.

Chapter 54
Serial Advanced Technology Attachment PHY (SATA PHY)

54.1 Overview

The Serial-ATA PHY is an ultra low-power SATA physical layer that complies with *Serial ATA*, Revision 2.5.

54.1.1 General Product Description

The SATA2 PHY is a complete mixed-signal IP solution designed to implement SATA connectivity in a System-on-Chip (SoC) design targeted to a specific fabrication process.

The SATA2 PHY provides a unique combination of:

- Small area
- Low power
- High performance
- Testability and characterization (diagnostic) features

54.1.1.1 System Overview

Each SATA2 PHY lane takes a 10- or 20-bit input and produces a serial output at 10 or 20 times the input word rate, respectively.

The transmitted data is synchronous to the local refclk. The receiver is clock- forwarded (source-synchronous) and the received data and its accompanying clock are synchronous to refclk at the far end of the link. Synchronicity with refclk depends on the system application.

54.1.2 Features

The SATA2 PHY provides the following features:

- Data rates of 1.5/3.0 Gbps, selectable per transceiver
- Clock module:
- Pin-programmable MPLL bandwidth from 2.5-10 MHz in 1.25-MHz increments
- Selectable use of an alternative reference clock delivered from the ASIC
- Flexible, glitchless clock outputs to ASIC: word rate, 1/2 word rate, and keep-alive clock options buffered from RefClk
- Power on or off and suspension of RefClk
- Flexible baud rate: RefClk ratio. Includes all integers greater than 3 (excluding 6, 7, and 11) and less than 67, and all even integers up to and including 130.
- RefClk frequencies of 25 MHz or 50-156.25 MHz. Common RefClk frequencies of 100 MHz, 125 MHz, and 156.25 MHz are supported.
- Generation of low jitter spread-spectrum clock (0.5% downspread at 31.5 KHz)
- Transmit path:
  - Multiple power-down modes
  - Differential Tx amplitude with less than ± 10% variation across temperature, voltage, and process
  - Pin-programmable transmit level
  - Pin-programmable attenuation of {8, 9, 10, 12, 14, 16} / 16 of full scale
  - Pin-programmable Tx boost of 0-5.75 dB in increments of ~ .37 dB
  - 10- or 20-bit wide ASIC interface
  - Per-lane word clock (in addition to word clock from MPLL) for maximum flexibility
  - Out of Band (OOB) signaling
  - Latency of 22-24 bit times from when the 10-bit data is sampled to when the first bit of the word is transmitted serially
- Receive path:
  - Pin-programmable equalization of .5-4 dB in .5-dB increments
  - Programmable Rx clock data recovery (CDR) based on a digital PLL (DPLL) for robust operation despite independently spread-spectrum references
  - Rx CDR that can tolerate run lengths of thousands of bits
  - Pin-programmable loss of signal (LOS) threshold
  - Pin-programmable ACJTAG hysteresis level
  - Multiple power-down modes
  - Rx bandwidth of 2.5 GHz
  - 10- or 20-bit wide ASIC interface
  - Clock forwarding
  - OOB signaling

- Latency of 22-31 bit times from when the last bit of a 10-bit word is received to when the 10-bit word is available on the ASIC interface
- Tx and Rx termination impedances that are automatically calibrated to match the external reference resistance

Test Features

The SATA2 PHY provides the following test features.

- Integrated test features:
  - Combinatorial loopback for full testing of the ASIC/IP interface with the ASIC's native testing methodology
  - Burn-in mode to toggle most internal modes without using clocks or controls
  - IDDQ test mode
- Loopback:
  - Tx-to-Rx serial analog loopback for wafer probe only
  - Tx-to-Rx serial digital loopback
- Byte Error Rate Testing (BERT) independent per lane:
  - 7th- and 15th-order polynomial pattern generation and recognition
  - Generation of simple test patterns
  - Byte error counting from polynomial patterns or simple test patterns
  - Voltage margining with 10-bit resolution, synchronous or asynchronous operation
  - Phase margining with UI/512 resolution, synchronous or asynchronous (when number of lanes is greater than one) operation
  - Combined 2D margining
- High resolution scope per Rx signal pair:
  - Acquisition of eye from patterns of known periodicity
  - Acquisition and analysis of signals of modest periodicities (< 1,024 bytes)
- Analog DC testing:
  - 10-bit A/D converter
  - Selection and measurement of any individual Rx and Tx termination resistor
- Limit testing that enables vector-only tests, which pass results in a non-trivial limit range:
  - High/low limits
  - Whether register read is within limits
  - Whether difference between register readback values is within limits

54.1.3 Block Diagram

The figure below shows a high-level SATA2 PHY block diagram.

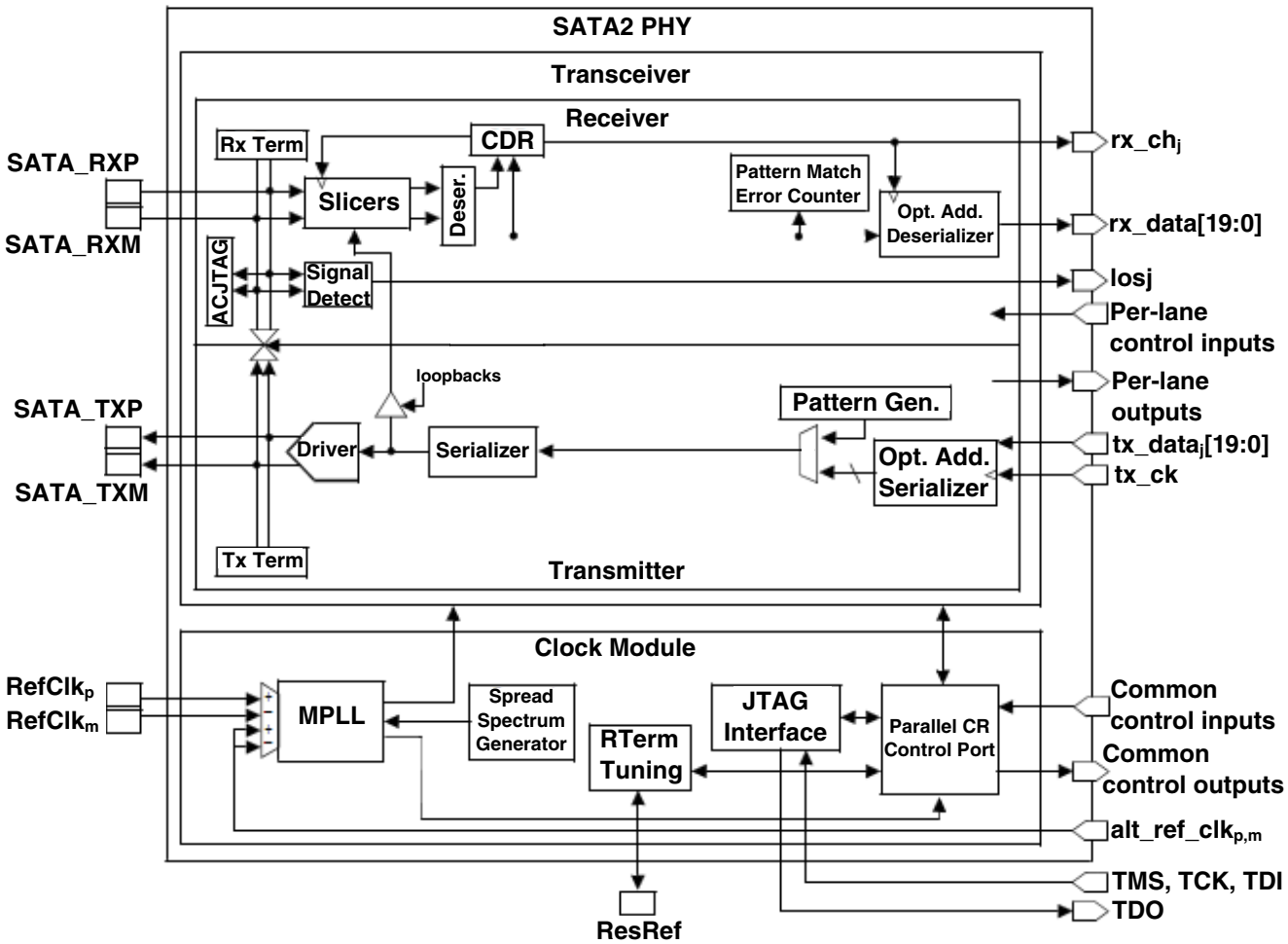


Figure 54-1. SATA2 PHY Block Diagram

54.1.4 Block Descriptions

This section describes blocks in the SATA2 PHY.

Transceiver Block - The Transceiver block is replicated per lane. This section describes blocks in the transceiver.

Receiver Block - This block converts the incoming, high-speed differential serial link to 10- or 20-bit parallel data and recovers the clock.

- **Rx Termination Block** - This block produces single-ended termination impedances of ~50 ohms to ground, differential ~ 100 ohms.

- **Rx Signal Detect** - This block detects the presence of a signal on the Rx signal pair. In a SATA implementation, this block detects OOB signals.

- **ACJTAG (Boundary Scan) Interface** - This block contains the logic for doing industry-standard (1149.1-2001 and 1149.6-2003) connectivity testing.

- **Rx Slicers** - This block converts the receive signal from a high-bandwidth analog signal to a sequence of 1s and 0s sampled when directed by the Clock and Data Recovery (CDR) block.

- **Deserializer** - The deserializer converts the output of the slicers to a 10-bit wide (1 coded byte) format. The deserializer uses COMMA characters to select the correct word alignment.

- **Clock and Data Recovery** - The CDR block comprises both a digital PLL (DPLL) and an analog PLL (APLL). The DPLL comprises phase detectors and a digital loop filter, which combine to produce a selected output phase. The APLL generates and filters the clock phase selected by the DPLL. The generated output clock feeds the slicers with multiple phases, while another output is divided down to become the word rate (baud rate / 10) recovered clock.

- **Rx Pattern Match and Error Count** - To identify errors during characterization of the channel quality, this block determines whether each byte of the received data sequence matches one of a few pseudo-random sequences. Detected byte errors are counted and available in the Pattern Matcher Control register (see lane0 Memory Map/Register Definition).

- **Optional Additional Deserializer** - When enabled by the assertion of the wide_xface control, this block doubles the Rx datapath width (from 10 bits to 20 bits) and reduces the Rx_Ck rate to baud_rate/ 20. When wide_xface is deasserted, only bits [9:0] of the RxData bus are used; Rx_Ck runs at baud_rate / 10. When the half_rate signal is asserted, clock frequencies are halved, but still based on the true baud rate as previously described.

Transmitter Block - This block converts outgoing, parallel 10- or 20-bit data to a high-speed, differential serial link.

- **Tx Termination Block** - This block produces single-ended termination impedances of ~ 50 ohms to the I/O supply or ground.

- **Tx-to-Rx Analog Serial Loopback Switch** - This switch establishes a serial loopback connection between Tx and Rx differential signals for wafer test only. This analog loopback encompasses the complete analog signal path.

- **Tx-to-Rx Digital Serial Loopback Switch** - This switch establishes a serial loopback connection between the Tx and Rx circuits. This digital loopback encompasses most of the signal path, skipping the actual transmit driver and the first stage of the receiver.

- **Optional Additional Tx Serializer** - When enabled in Wide (20-bit) mode by assertion of the wide_xface control, this block reduces the width of the Tx datapath from two words to one word. In Wide mode, the Tx_Ck rate is expected to be 1/20th the baud rate. When the wide_xface control is deasserted (10-bit mode), only bits [9:0] of the TxData bus are used. In either 10- or 20-bit mode, this block converts Tx data from the Tx_Ck domain to the internal "baud / 10" domain. Tx_Ck is derived by buffering the cko_word or tx_cko_word clock. When half_rate is asserted, the clock frequencies are halved, but still based on the true baud rate as previously described.

- **Tx Pattern Generator** - This block produces one of a few pseudo-random sequences that can be easily matched in a receiver as part of the built-in Byte Error Rate Testing (BERT) function.

- **Tx Serializer** - This block serializes data, converting 10 bits at the word rate to 1 bit at the baud rate.

- **Tx Driver** -When fully enabled, this block drives a serialized signal with the programmed signal level and de- emphasis.

Clock Module - This section describes blocks in the Clock module.

- **Multiplying PLL** - The Multiplying PLL (MPLL) block converts the provided reference clock into the "baud / 2" rate required. This high-speed differential clock is buffered internally through an array of transceivers. The cko_word clock is produced at 1/10th or 1/20th the MPLL baud rate.

- **Spread Spectrum Generator** - This block generates the appropriate clock offsets for SATA spread spectrum support.

- **RTerm Tuning** -This block uses an external resistor (ResRef) as reference to determine the proper calibration setting for centering the Rx and Tx termination resistances at 50 ohms.

- **JTAG Interface** - This block enables test mode programming of internal test features.

NOTE

To use the SATA2 PHY in product applications, JTAG transactions are not required. To use the provided automatic test equipment (ATE) vectors and diagnostic capabilities, the JTAG interface is required.

54.2 External Signals

The table found here describes the external signals of SATA PHY.

Table 54-1. SATA PHY External Signals

Signal	Description	Pad	Mode	Direction
SATA_PHY_RX_N (RX_N)	Negative receive signal	SATA_RXM	No Muxing	I
SATA_PHY_RX_P (RX_P)	Positive receive signal	SATA_RXP	No Muxing	I
SATA_PHY_TX_N (TX_N)	Negative transmit signal	SATA_TXM	No Muxing	O
SATA_PHY_TX_P (TX_P)	Positive transmit signal	SATA_TXP	No Muxing	O

54.3 Functional Description

This section describes SATA2 PHY functions.

54.3.1 Power Controls

This section describes power-down controls available to the ASIC.

54.3.1.1 Tx Power Controls

The table below provides the recommended Tx power state mappings.

Table 54-2. Recommended SATA Tx Power State Mappings

tx_en[2:0]	SATA
000 (OFF)	Disabled
001 (CM)	Slumber
010 (CM_CLK)	Partial
011 (ON)	Enabled/OOB

For tx_en[2:0] settings, see Per-Transceiver Control and Status Signals.

The table below lists all possible state changes. The SATA2 PHY supports only changes that are indicated as valid.

Most changes are immediate (only logic and signal delay times). Changes that are not immediate are complete when tx_done toggles, as noted in the table below. The tx_done signal also toggles due to the assertion of tx_clk_align.

If the tx_ck input changes in phase or frequency, the internal transmit clock must be resynchronized to the tx_ck input before driving data. This resynchronization occurs automatically during state transitions. If this resynchronization is required outside these transitions, resynchronization can be requested by asserting the tx_clk_align input.

Table 54-3. Transmitter State Transitions

From State	To State	Valid Change	tx_done Toggle	Clock Resync
OFF	CM	Yes	No	No
OFF	CM_CLK	Yes	Yes	Yes
OFF	ON	Yes	Yes	Yes
CM	OFF	Yes	No	No
CM	CM_CLK	Yes	Yes	Yes
CM	ON	Yes	Yes	Yes
CM_CLK	OFF	Yes	No	No
CM_CLK	CM	Yes	No	No
CM_CLK	ON	Yes	No	No
ON	OFF	Yes	No	No
ON	CM	Yes	No	No
ON	CM_CLK	Yes	No	No

54.3.1.2 Rx Power Controls

The receiver function is controlled through the rx_en, rx_pll_pwr_on, and rx_term_en pins. The successful change of the PLL power state (due to change in rx_pll_pwr_on) is signaled back to the ASIC by transitioning the rx_pll_state signal.

The LOS function continues to operate unless explicitly disabled. The table below provides the recommended Rx power state mappings.

Table 54-4. Recommended SATA Rx Power State Mappings

SATA	rx_en	rx_pll_pwr_on	rx_term_en
Disabled	0	0	0
Slumber	0	0	1
Partial	0	0/1	1
Enabled/OOB	1	1	1

or fastest exit from partial, set rx_pll_pwr_on to 1'b1. For lower power, set rx_pll_pwr_on to 1'b0.

When rx_en is set to 1'b0, rx_ck is disabled; therefore, either asynchronous logic or another clock source must be used to move between states during this period.

54.3.1.3 Clock Module Power Controls

The table below provides the MPLL power state settings.

Table 54-5. Recommended SATA Power State Mappings

SATA	mpll_pwron (PCLK State)
Disabled	0
Slumber	1
Partial	1
Enabled/OOB	1

The Clock module contains the MPLL and is the entry point for the SATA2 PHY reference clock. The MPLL can be powered down using this module, and the reference clock coming into the PHY can be suspended.

Before disabling the reference clock externally or suspending the clock via the mpll_ck_off pin, always power down the MPLL. The mpll_ck_off pin gates the reference clock on and off at the entry point to the SATA2 PHY. Before disabling the external reference clock, always set mpll_ck_off to prevent needless power consumption in the block that converts the reference clock to full logic levels. Setting mpll_ck_off disables

all clocks (even cko_alive). When powering down the MPLL, consider whether you want cko_word to be suspended or switched to the reference clock and if you want the cko_alive clock to be available.

54.3.1.4 Power-Up Sequences

This section describes various power-up sequences.

54.3.1.4.1 Powering Up the Chip (Initial Power-Up)

To perform initial chip power-up:

1. Power up the power supplies.

NOTE

If you use the power_good indicator to determine when the power supplies have all reached 80% of their respective nominal values, ensure that you power up the lowest supply first.

2. Without using a clock from the SATA2 PHY, enable the external reference clock and wait for it to become stable.

3. To propagate refclk to the MPLL, set mpll_ck_off to 1'b0.

NOTE

Before setting mpll_ck_off to 1'b0, set the mpll_ncy, mpll_ncy5, and mpll_prescale signals to the appropriate values.

4. Perform a PHY reset by either toggling reset_n or writing a 1 to the Reset register (see Reset Register) through the JTAG interface or Parallel CR Control port.

Upon writing the PHY reset bit in the reset register, the internal PHY reset is active immediately. Since the reset also affects the control register state machine, there will not be an acknowledgement of the write; that is, cr_ack will not be asserted.

NOTE

Diagnostic code should treat the *lack* of an acknowledgment of the write as a *successful* write; alternatively, it should treat the PHY *acknowledging* a write of the reset as a write *failure*. This is the opposite expectation of all other registers, where the lack is a failure and the *acknowledge* is successful. It is sufficient to wait 20 ref_clock cycles in order to determine that the acknowledgement has *not* occurred.

For information about the JTAG interface and the Parallel CR Control port, see Block Descriptions.

54.3.1.4.2 Powering Up the Clock Module

The following procedure assumes the chip is powered up and all blocks are in their power-down state. To power up the Clock module:

1. To start the MPLL, set the mpll_pwron signal to 1'b1.
2. Wait for op_done to transition.

The Tx and Rx can now be powered up as appropriate.

54.3.1.4.3 Powering Up the Tx

Assuming that the Clock module is powered up, tx_en[2:0] can be changed to any mode. Some transitions might require waiting for tx_done to transition before the transmitter is in the intended state.

54.3.1.4.4 Powering Up the Rx

The following procedure assumes that the Clock module is powered up.

To power up the Rx:

1. Turn on receiver terminations by setting rx_term_en to 1'b1.
2. Power up the Rx PLL by setting rx_pll_pwron to 1'b1.
3. Wait for rx_pll_state to go high.
4. Enable rx_ck and drive data out on the rx_data pins by setting rx_en to 1'b1.
5. Wait for rx_valid to indicate that the data is valid.

54.3.1.5 Power-Down Sequences

This section describes various power-down sequences.

54.3.1.5.1 Powering Down the Rx

To power down the Rx:

1. To disable the rx_ck and rx_data pins, set rx_en to 1'b0.
2. To power down the Rx PLL, set rx_pll_pwron to 1'b0.
3. Wait for rx_pll_state to go low.

54.3.1.5.2 Powering Down the Tx

To disable the transmitter, set tx_en[2:0] to the OFF or CM setting.

54.3.1.5.3 Powering Down the Clock Module

This procedure assumes the Tx and Rx are powered down.

To power down the Clock module:

1. If you want refclk to be output on cko_word while the MPLL is powered down, change cko_word_con to the refclk setting (3'b000).
2. To power down the MPLL, set mpll_pwron to 1'b0.
3. Wait for op_done to transition.
4. If refclk is to be suspended, or if mpll_ncy, mpll_ncy5, or mpll_prescale must be changed, first set mpll_ck_off to 1'b1.

54.3.2 Clock Module Operations

Figure 54-2 below shows the input and output clocks for one receiver/transmitter pair and the Clock module.

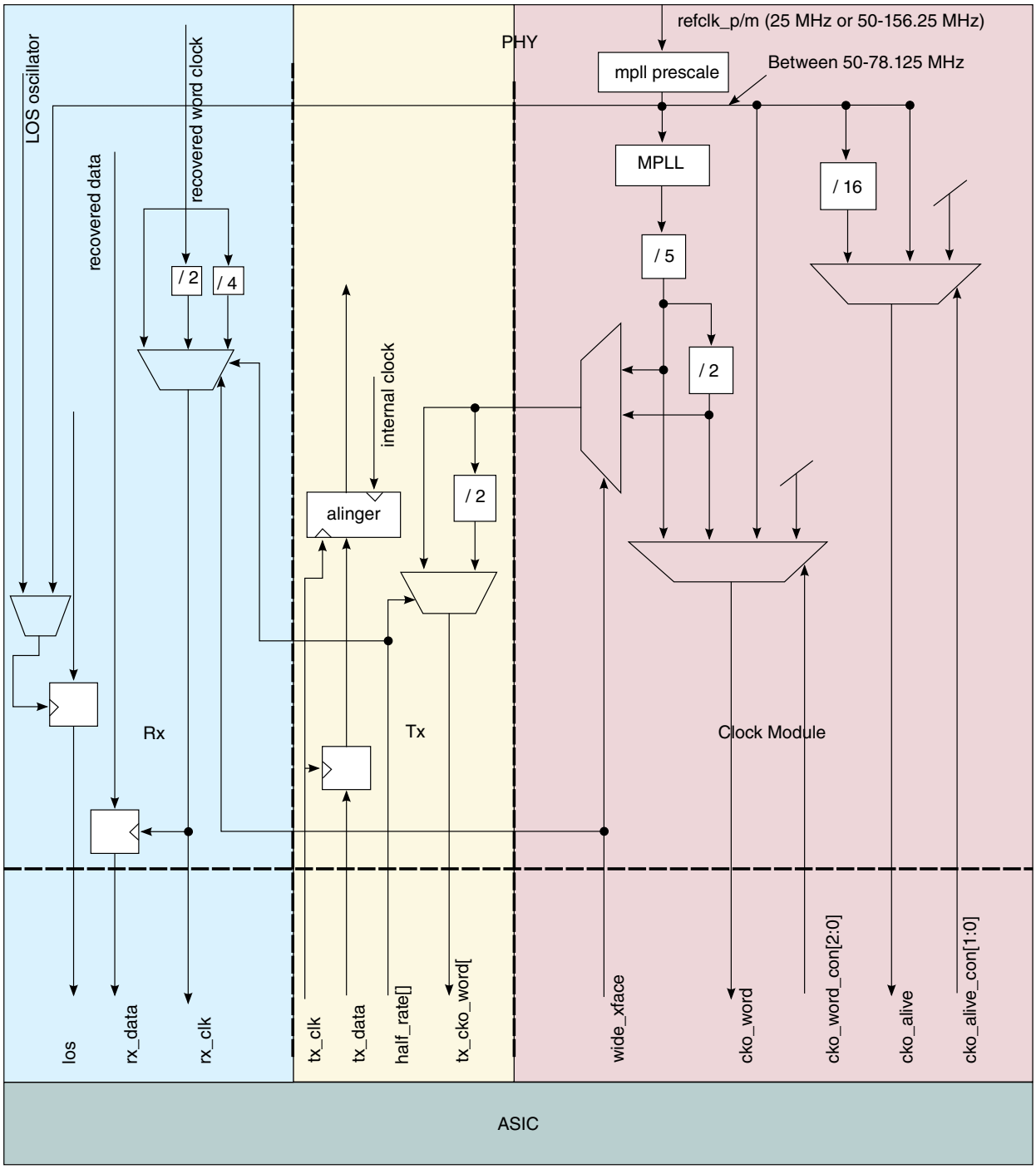


Figure 54-2. Clock interface

Note: tx_ck must be derived from tx_cko_word or cko_word.

54.3.3 Clock Inputs to the SATA2 PHY

The SATA2 PHY reference clock can be sourced from the ref_clk_p/m pads or the alt_ref_clk_p/m pins. The SATA2 PHY reference clock input has a frequency range of 25 MHz or 50-156.25 MHz.

Although the MPLL has a wide frequency range, there is a subset of specific input frequences that achieves the desired data rate. The reference clock is not used by the MPLL directly; instead, the clock passes through a prescaler that can pass the ref_clk through or divide/multiply the ref_clk frequency by 2. To guarantee that the MPLL input is in the range of 50-78.125 MHz, the prescaler must be set properly.

54.3.3.1 mpll_prescale[1:0]

The prescaler is a block that inputs the reference clock and outputs the clock used as input to the MPLL and the cko_word and cko_alive multiplexers.

The MPLL expects input clock (output from the prescaler) frequencies of 50-78.125 MHz. The table below provides some common settings. Set up the MPLL to operate at half the baud rate (except for SATA I where the MPLL operates at baud rate).

Valid refclk Range and Formulaic MPLL Settings provides a general formulaic approach for determining settings for less common scenarios. In the table below, the MPLL Divider column can be decoded into the appropriate mpll_ncy[4:0] and mpll_ncy5[1:0] settings using Table 54-8.

Table 54-6. mpll_prescale[1:0]: Sample Reference Clock Configurations

Reference Clock Frequency (MHz)	pll_prescale[1:0]	mpll_prescale Effect on ref clk	Input to MPLL (MHz)	MPLL Divider	Resultant Baud Rate (Gbps)
25	01	x 2	50	30	3
50	00	As is	50	30	3
60	00	As is	60	25	3
62.5	00	As is	62.5	24	3
75	00	As is	75	20	3
100	10	/ 2	50	30	3
120	10	/ 2	60	25	3
125	10	/ 2	62.5	24	3
150	10	/ 2	75	20	3

54.3.3.2 Valid refclk Range and Formulaic MPLL Settings

This section presents a formulaic approach for setting the MPLL values.

As mentioned in mpll_prescale[1:0], the MPLL input must be in the range of 50-78.125 MHz. Frequency doubling and halving is available at the refclk input. The following table lists all valid refclk ranges and the appropriate mpll_prescale[1:0] settings to place refclk in the proper range. This list includes valid refclk frequencies, but there is no guarantee that a particular refclk frequency can be used to create the intended baud rate.

Table 54-7. Valid refclk Frequencies

refclk Frequency Range (MHz)	mpll_prescale[1:0]	Effect on refclk Frequency	Clock Input to MPLL (MHz)
25	01	Doubled	50
50-78.125	00	No effect	50-78.125
100-156.25	10	Halved	50-78.125

To create the intended baud rate, the clock input to the MPLL (as determined from Table 2-5 on page 31) must evenly divide half the intended baud rate (or the baud rate in Half-Rate mode). The formula for Full- Rate mode is:

MPLL Divider = 0.5 x (baud rate) / (clock input to MPLL)

The formula for Half-Rate mode is:

MPLL Divider = (baud rate) / (clock input to MPLL)

The MPLL Divider value can then be used to determine the values for the mpll_ncy5 and mpll_ncy buses, as listed in the following table.

NOTE

The preceding formulae must yield a valid integer. If the result is not an integer, a refclk frequency that produces a valid integer must be used. The table below lists valid divider values.

Table 54-8. MPLL Divider Settings

MPLL Divider	mpll_ncy[4:0]	mpll_ncy5[1:0]
4	00000	00
5	00000	01
8	00001	00
9	00001	01
10	00001	01
12	00010	00
13	00010	01

Table continues on the next page...

Table 54-8. MPLL Divider Settings (continued)

MPLL Divider	mpll_ncy[4:0]	mpll_ncy5[1:0]
14	00010	10
15	00010	11
16	00011	00
17	00011	01
18	00011	10
19	00011	11
20	00100	00
21	00100	01
22	00100	10
23	00100	11
24	00101	00
25	00101	01
26	00101	10
27	00101	11
28	00110	00
29	00110	01
30	00110	10
31	00110	11
32	00111	00
33	00111	01
34	00111	10
35	00111	11
36	01000	00
37	01000	01
38	01000	10
39	01000	11
40	01001	00
41	01001	01
42	01001	10
43	01001	11
44	01010	00
45	01010	01
46	01010	10
47	01010	11
48	01011	00
49	01011	01
50	01011	10
51	01011	11
52	01100	00

Table continues on the next page...

Table 54-8. MPLL Divider Settings (continued)

MPLL Divider	mpll_ncy[4:0]	mpll_ncy5[1:0]
53	01100	01
54	01100	10
55	01100	11
56	01101	00
57	01101	01
58	01101	10
59	01101	11
60	01110	00
61	01110	01
62	01110	10
63	01110	11

54.3.3.3 Clock Module Power Control

For information about the Clock module power controls, see Clock Module Power Control.

54.3.3.4 Presence of refclk Signal

The reference clock must be present during all operational modes, except when the MPLL is powered down and mpll_ck_off has been set to 1'b1.

For more information, see Power-Up Sequences and Power-Down Sequences.

54.3.3.5 Power-On Reset

The SATA2 PHY asserts its internal power-on reset (POR) whenever the voltage level of the high supply has not yet reached the initial "power valid" trip point at ~70-75% of the rated supply voltage.

After a POR is asserted, to prevent accidental reactivation of the POR, the trip point is reduced to ~65-70% of the rated supply voltage. Reactivation of a POR causes the SATA2 PHY to go into a reset state again.

Note that deassertion of a POR causes a normal transition out of reset. During power-up, constraints on the state of the control inputs must be maintained-meaning that the POR mechanism in the SoC must ensure that control inputs are set to the intended states.

54.3.3.6 Resistor Calibration

During a reset or at the request of the ASIC via the rtune_do_tune pin, a resistor calibration occurs. When the operation is complete, the op_done pin is transitioned.

Resistor calibration consists of learning which state of the internal Resistor Calibration register causes an internal, digitally trimmed calibration resistor to best match the impedance applied to the ResRef pin. The calibration register value is then supplied to all Tx and Rx termination resistors.

During the calibration process (for a few tens of microseconds), up to 0.3 mW can be dissipated in the external ResRef resistor. At other times, no power is dissipated by the ResRef resistor.

54.3.4 Tx Operations

This section describes transmitter settings and operations.

54.3.4.1 Recommended Tx Settings

The following table outlines the recommended transmitter settings, and the sections that follow describe each setting in detail.

Table 54-9. Recommended Tx Settings

Application	tx_atten[2:0] (Short/ Medium/Long)[1]	tx_boost[3:0] (Short/ Medium/Long)[1]	tx_lvl[4:0] (Short/Medium/ Long)[1]	x_edgerate[1:0]
SATA 1i	011	N/A	00110	01
SATA 1m	011	N/A	00110	01
SATA 1x	N/A	1001	00110	01
SATA 2i	100	N/A	10001	00
SATA 2m	100	N/A	10001	00
SATA 2x	000	1001	10001	00

1. Short/medium/long are typical values based on 5-cm, 23-cm, and 48-cm trace lengths on FR4 material. These values are provided as guidelines. It is recommended that the final setting be based on the characteristics of the actual serial channel.

54.3.4.2 Tx Amplitude Control

The transmitter has a programmable boost, level, and attenuation. The transmitter's full-scale amplitude is a function of the selected level (tx_lvl), attenuation (tx_atten), and boost (tx_boost).

The actual peak-to-peak differential amplitude is the Tx amplitude multiplied by the attenuation multiplier:

$$\text{Tx amp}_{\text{(p-p differential near-end)}} = \text{attenuation multiplier} \bullet \frac{1.24}{63.5} \left[\left(48 + 0.5 \left(\text{tx\_lvl[4:0]} \right) - V \right] \right]_{\text{, p-p differential near-end}}$$

Using a larger transmit amplitude provides a larger eye to a far-end receiver (at the expense of a slight increase in power).

For tx_lvl[4:0] settings, see Common Signals.

54.3.4.3 Tx Boost Control

The transmitter can be programmed to provide boost (pre-emphasis/de-emphasis). Boost is achieved by reducing the drive level of a non-transition bit with respect to a transition bit.

The amount of boost the transmitter supplies is programmed through the tx_boost[3:0] pins as follows:

$$\text{boost} = -20\log\left(1 - \frac{\text{txboost[3:0]} + 0.5}{32}\right) \text{ db}$$

except that setting tx_boost[3:0] to 4'b0000 actually produces 0 db of boost. This boost control produces results up to 5.75 dB in increments of ~0.37 dB. These pins can be transitioned asynchronously. The best boost setting is channel dependent. For lossy channels, the maximum boost setting is appropriate; for low- loss channels, lower settings provide a better eye. Table 54-9 provides the recommended boost settings for short, medium, and long trace lengths. If lossy channels are expected in the application, setting the boost to its maximum value is appropriate. If more information is known about the channel, a good rule is to take the loss at Nyquist, subtract 3 dB, and find the tx_boost

setting that most closely achieves this value. For example, if the worst-case loss at Nyquist is 6.5 dB, a good setting for tx_boost is one that achieves 3.5 dB of boost. From the preceding equation, the tx_boost setting is 4'b1010.

54.3.4.4 Tx Far-End Amplitude

The transmitter far-end amplitude is a function of boost, level, and attenuation.

The actual far-end differential amplitude can be defined as the most commonly transmitted level:

$$\left(-\frac{\text{boost}_{db}}{\text{attenuation factor}}\right) = \text{atten}_{multiplier} \cdot 10^{20}$$

$$\text{Tx amp}_{\text{(differential far-end)}} = \frac{\text{attenuation}}{\text{factor}} \cdot \frac{1.24}{63.5}\left[\left(48 + 0.5\left(\text{tx\_lvl[4:0]}\right) - V\right]_{\text{, differential far-end}}\right.$$

With proper boost settings, the actual far-end differential amplitude is approximately equal to the far-end peak-to-peak differential amplitude.

54.3.4.5 Tx Edge Rate Control

The following table describes the edge rate control settings, which enable the SATA2 PHY to meet the edge rate requirements of all SATA variants.

Table 54-10. tx_edgerate Settings

tx_edgerate[1:0]	Value
00	Fast edge rate
01	Medium edge rate
10	Slow edge rate
11	Reserved

54.3.5 Rx Operations

This section describes receiver operations.

54.3.5.1 Recommended Rx Settings

The table below describes the recommended Rx settings.

Table 54-11. Recommended Rx Settings

Application	rx_eq_val[2:0]	los_lvl[4:0]	rx_dpll_mode[2:0]
SATA1i	101	10000	011 with SS 001 without SS
SATA1m	101	10000	011 with SS 001 without SS
SATA1x	101	11010	011 with SS 001 without SS
SATA2i	101	10010	011 with SS 001 without SS
SATA2m	101	10010	011 with SS 001 without SS
SATA2x	101	11010	011 with SS 001 without SS

54.3.5.2 Loss of Signal Detection

Flexible LOS detection is provided on a per-lane basis. The per-lane output of the LOS detection is provided through the LOS pins. LOS detection is controlled by filtering the raw LOS signal using the los_ctl pins and setting the LOS threshold.

Table 54-12 describes the los_ctl settings.

Table 54-12. los_ctl Settings

los_ctl[1:0]	Description
00	Disabled
01	Reserved
10	Filtering for SATA to enable OOB processing. The LOS signal is synchronous to the output of the prescaler (available on cko_alive or cko_word).
11	Heavy filtering. The LOS signal is synchronous to the prescaler's output (available on cko_alive or cko_word). Heavy filtering needs longer periods of inactivity or activity before indicating an LOS event because of slower response time to chatter on the receiver lines.

As defined in the SATA specification, OOB signaling is accomplished by measuring the gaps between bursts of energy. An energy burst is defined to be 160 UI long (107 ns long) at SATA I rates. The burst consists of repeating ALIGN primitives (k28.5, d10.2, d10.2, d27.3).

Table 54-13. Words Used in ALIGN Primitive

ALIGN words	+ Disparity	- Disparity
k28.5 (COMMA)	1 1 0 0 0 0 0 1 0 1	0 0 1 1 1 1 1 0 1 0
d10.2 (NYQUIST)	0 1 0 1 0 1 0 1 0 1	0 1 0 1 0 1 0 1 0 1
d27.3	1 1 0 1 1 0 0 0 1 1	0 0 1 0 0 1 1 1 0 0

The ALIGNs during OOB signaling are always sent at SATA I rates. The communication is realized by the length of the gap between bursts. The following three tables describe the OOB gap-sensing limits for COMWAKE, COMSET/COMINIT, and COMSAS, respectively.

Table 54-14. OOB Gap-Sensing Limits for COMWAKE

Gap Length	Decision
< 55 ns	Must not indicate a gap
55-101 ns	Can indicate COMWAKE
101-112ns	Must indicate COMWAKE
112-175ns	Can indicate COMWAKE
> 175 ns	Must not indicate there was a gap

Table 54-15. OOB Gap-Sensing Limits for COMRESET/COMINIT

Gap Length	Decision
< 175 ns	Must not indicate a gap
175-304 ns	Can indicate COMRESET/COMINIT
304-336 ns	Must indicate COMRESET/COMINIT
336-525 ns	Can indicate COMRESET/COMINIT
> 525 ns	Must not indicate there was a gap

Table 54-16. OOB Gap-Sensing Limits for COMSAS

Gap Length	Decision
< 525 ns	Must not indicate a gap
525-911.7 ns	Can indicate COMSAS
911.7-1,008 ns	Must indicate COMSAS
1,008-1,575 ns	Can indicate COMSAS
> 1,575 ns	Must not indicate there was a gap

For OOB detection/transmission, to synchronously transfer the LOS signal to the ASIC, the SATA2 PHY relies on a clock that is proportional to the reference clock. If the reference clock frequency is 25 MHz, it is multiplied by 2-50 MHz; if the reference clock frequency is 100 MHz, it is divided by 2-50 MHz. In addition, the LOS signal is preprocessed. Therefore, based on the results, it is recommended that the decision points indicated in Table 2-16 be used for detecting OOB signals.

Table 54-17. Recommended Decision Points (Assumes 25-MHz, 50-MHz or 100-MHz Reference Clock)

LOS Length (20-ns Periods)	Decision	True Limits (ns)
0-2	Invalid gap length	< 80
3-6	COMWAKE	70-150
7-10	Invalid gap length	140-230
11-22	COMINIT/COMRESET	220-470
22-28	Invalid gap length	460-590
29-75	COMSAS	580-1,530
>=76	Invalid gap length	>=1,520

NOTE

Recommended decision points assume a reference clock of 25 MHz, 50-MHZ, or 100 MHz.

To measure/count the LOS length, it is recommended that you use cko_alive with cko_alive_con[1:0] set to 2'b01. With this setting, the cko_alive frequency will be 50 MHz .

54.3.5.3 rx_dpll_mode[2:0]

The table below describes the rx_dpll_mode[2:0] settings. At each point in time, the actual gain setting is the larger of the gains indicated by the fast startup (if in use) and rx_dpll_mode[1:0]. The values of Phase

Update Gain (PHUG) and Frequency Update Gain (FRUG) correspond to the proportional and integral gains of the DPLL. Higher gains in the PHUG and FRUG settings cause an increase in jitter.

Table 54-18. rx_dpll_mode Settings

rx_dpll_mode[2:0]	PHUG	FRUG	fast_startup	Frequency Tolerance (ppm)
000	1	1	None	780
001[1]	2	2	None	780

Table continues on the next page...

Table 54-18. rx_dpll_mode Settings (continued)

rx_dpll_mode[2:0]	PHUG	FRUG	fast_startup	Frequency Tolerance (ppm)
101	1	4	None	6,250
011[2]	2	4	None	6,250
100	Reserved			
101				
110				
111				

1. Recommended setting (without Spread Sectrum (SS))
2. Recommended setting (with Spread Sectrum (SS))

54.3.5.4 Rx Equalizer Settings

The SATA2 PHY provides an Rx equalizer that can be programmed using the rx_eq_val[2:0] bits. The table below lists the approximate boost that each setting provides.

Table 54-19. Rx Equalizer Settings

rx_eq_val[2:0]	Boost (dB)
000	0.5
001	1.0
010	1.5
011	2.0
100	2.5
101	3.0
110	3.5
111	4.0

These pins can be transitioned asynchronously, but a dynamic change to these pins during data transmission might result in bit errors. Typically, these settings do not change during operation. Although the specification does not define receive equalization, a more robust link can be achieved by providing some equalization. A good general setting is to set the equalizer to approximately 3 dB of boost-suitable for many applications.

54.4 Control Registers

SATA2 PHY test features are controlled by registers that are accessible through the JTAG interface. Each register is a maximum of 16 bits wide.

Because some registers contain different fields, when changing a single field in a register, use a read-modify-write approach.

54.4.1 Register Fields

Some fields are single bits while other fields encompass the entire contents of the containment register. Some registers have Xs as reset values.

These registers are categorized as follows.

Registers That Reflect Inputs From the ASIC Interface

The values of these inputs are unknown on reset; therefore, the register's reset value is indeterminate.

These registers include:

- lane0.tx_stat
- lane0.rx_stat
- clock.freq_stat
- clock.ctl_stat
- clock.lvl_stat
- clock.creg_stat

Registers That Reflect Results of Operations Not Initialized on Reset

These registers might have actual Xs on part reset. However, those bits have no meaning until the associated function is enabled or initialized; until then, discard the read results of these register bits.

These registers include:

- lane0.out_stat
- lane0.pm_err
- clock.crcmp_stat
- clock.scope_count
- clock.adc_out

When reading a register, mask the result to view only the bits with which you are concerned. Correct masking disables the Xs from propagating through your logic.

54.4.1.1 Field Properties

The property string associated with each field is a concatenation of the applicable attributes listed in the table below. For example, the most common field property is RWCr, which denotes readable, writable, and subject to a reset.

Table 54-20. Register Attributes

Property	Description
R	Readable. A register read returns the current value of the register itself (if it is also writable) or the state to which the register is attached.
R2	Read operation on this register is pipelined. Two reads are required to get "current" value.
W	Writable
V	Volatile (that is, the value can change at any time)
I	Value is in signed integer format (2's complement).
Cr	Register is subject to being returned to its reset value on a reset.

In addition, each field requires two numeric parameters to specify the field's location in the register: a width (in bits) and an offset (in bits) from the register's least-significant bit (LSB).

54.4.1.2 Field Names

Fully specified field names consist of the concatenation of a register name and a field name, as in *register.field*.

Fields in registers comprising a single field (for example, clock.crcmp_lo_range.crcmp_lo_range) are typically named without the concatenation (for example, clock.crcmp_lo_range). In these cases, the single field name always matches the last portion of the register name.

54.4.1.3 Read/Modify/Write Operations

To modify one field in a register containing multiple fields, it is generally necessary to read the register, modify the target field, then write the entire result.

One approach to writing control registers is to monitor the contents of each field; this approach does not require read-modify-write operations. The read is not necessary if the current register value is known (reset value at the time of the reset updated with any subsequent modifications to values of other fields in the register). Therefore, volatile control fields such as DPLL PHASE are never combined in the same register with other control fields.

Methods that do not maintain this information must work by reading a register, modifying only the field being changed, then writing the register.

54.5 Timing and Specifications

54.5.1 SATA2 PHY Implementation-Specific Timing

The table below describes the timings for a standalone SATA2 PHY.

Table 54-21. SATA2 PHY Implementation-Specific Timing

Name	Description	Value
Transmit latency	Measured from when the 10-bit data is sampled to when the first word bit is serially transmitted.	22-24 bit times
Transmit latency	Measured from when the 10-bit data is sampled to when the last word bit is serially transmitted.	• 10-bit wide interface: 31-33 bit times • 20-bit wide interface: 41-43 bit-times
Receive latency	Measured from when the first bit of a 10-bit word is received to when the 10-bit word is available to be sampled at the ASIC interface. Alignment can be turned off.	• 10-bit wide interface: 31-40 bit times • 20-bit wide interface: 41-50 bit times
Receive latency	Measured from when the last bit of a 10-bit word is received to when the 10-bit word is available to be sampled at the ASIC interface. Alignment can be turned off.	22-31 bit times
refclk spin-up latency	Time from deassertion of a reset (assuming mpll_ck_off is deasserted) or deassertion of mpll_ck_off to when the reference clock is available for the MPLL and cko_alive.	< 1.65 µs (or 31 µs if refclk doubler is used)
MPLL lock latency	Time from assertion of mpll_pwron (assuming refclk is on and stable) to when the MPLL is ready and cko_word begins toggling.	< 41 µs
Resistor tuning	Time to perform a resistor tune	2 µs
Receive PLL lock latency	Time from assertion of rx_pll_pwron (assuming the MPLL is ready) to when rx_pll_state is asserted. Does not require incoming data transmission.	9 µs
Receive CDR lock time	Time from end of loss of signal or assertion of rx_en to assertion of rx_valid. These are worst- case values; nominal lock times might be shorter. Assuming 30 kHz minimum frequency spread spectrum clocking. Non-spread spectrum receiver lock will be significantly shorter.	37 µs
Transmit enable latency	Time from tx_en change to serial lines leaving common mode.	• CM to ON: 300 ns • CM_CLK to ON: Same as transmit latency
LOS detection latency	For information about loss of signal detection, see Loss of Signal Detection.	20 ns

54.5.1.1 Synchronous Tx Inputs

The Tx input, tx_data, is clocked into the Tx block at the rate of either baud / 10 or baud / 20 (based on the wide_xface setting) with a source-synchronous clock, tx_ck (that must be derived from either tx_cko_word or cko_word).

The figure below shows timing for the Tx inputs.

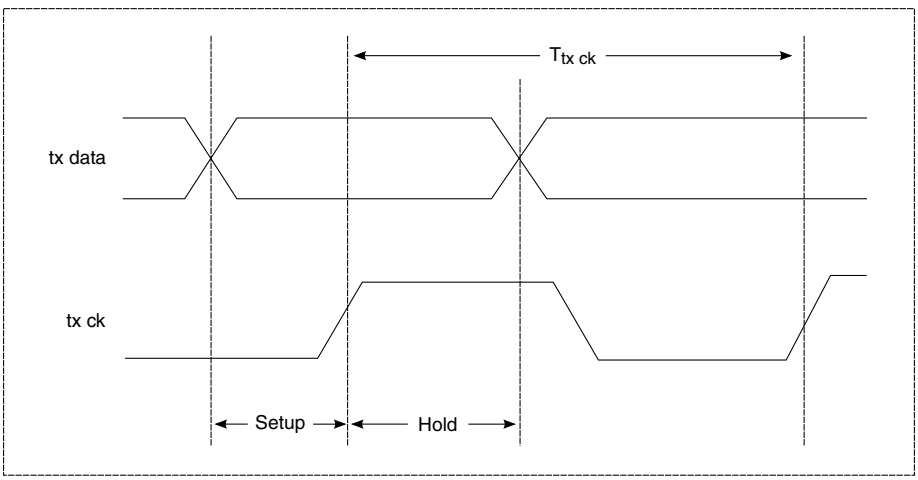


Figure 54-3. Parallel Input Tx Timing

54.5.1.2 Synchronous Rx Outputs

The Rx output data, rx_data, is generated with the rising edge of rx_ck (that has a Trx_ck of either 10 or 20 baud based on wide_xface). The data is stable based on the rising edge of this clock.

The figure below shows timing for the Rx outputs.

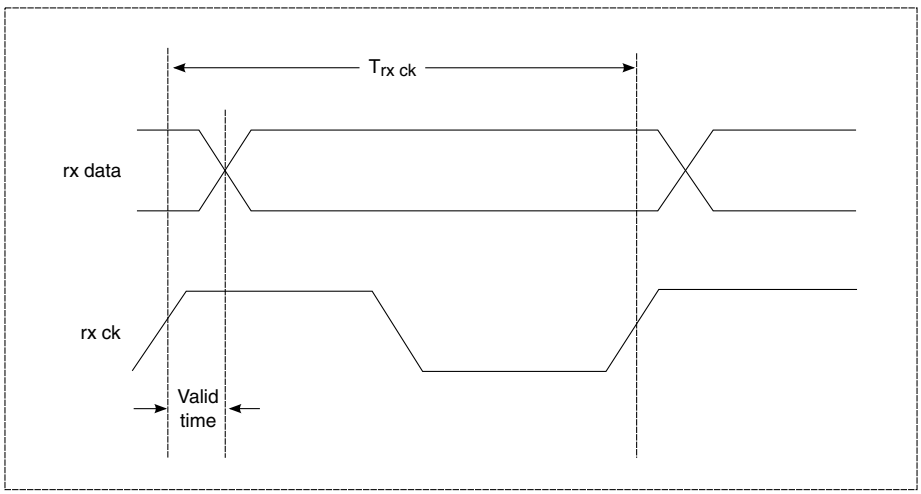


Figure 54-4. Parallel Output Rx Timing

54.5.1.3 Asynchronous Tx and Rx I/O

There are no static timing requirements for inputs or outputs noted as asynchronous in Table 1. The inputs take a short time to propagate to the associated circuitry that they control. Signals that do not have restrictions specified in Table 1 have no high or low pulse-width requirements. All outputs must be either synchronized before using or sampled appropriately for the pin's function.

54.5.1.4 Control Register Bus Interface

Timing of the Parallel Control Register bus is accomplished with a standard four-phase asynchronous handshake using the four CR request lines and the cr_ack signal.

For information about using this interface, see Parallel CR Control Port Testing.

Because the interface is asynchronous, it does not need to be timed with a static timing tool. There is zero setup time from cr_data_in[15:0] to the rising edge of cr_cap_{addr, data}. There is zero hold time from the rising edge of cr_ack to cr_data_in[15:0].

54.5.1.5 JTAG Interface Timing

The JTAG interface complies with interface pin timings as defined in the JTAG specification; therefore, you should not have timing issues interfacing this port with other internal or external JTAG implementations.

The tck clock signal has a maximum frequency of 35 MHz, but no minimum frequency. This clock can be stopped at any point without loss of state. The figure below shows timing for the JTAG interface.

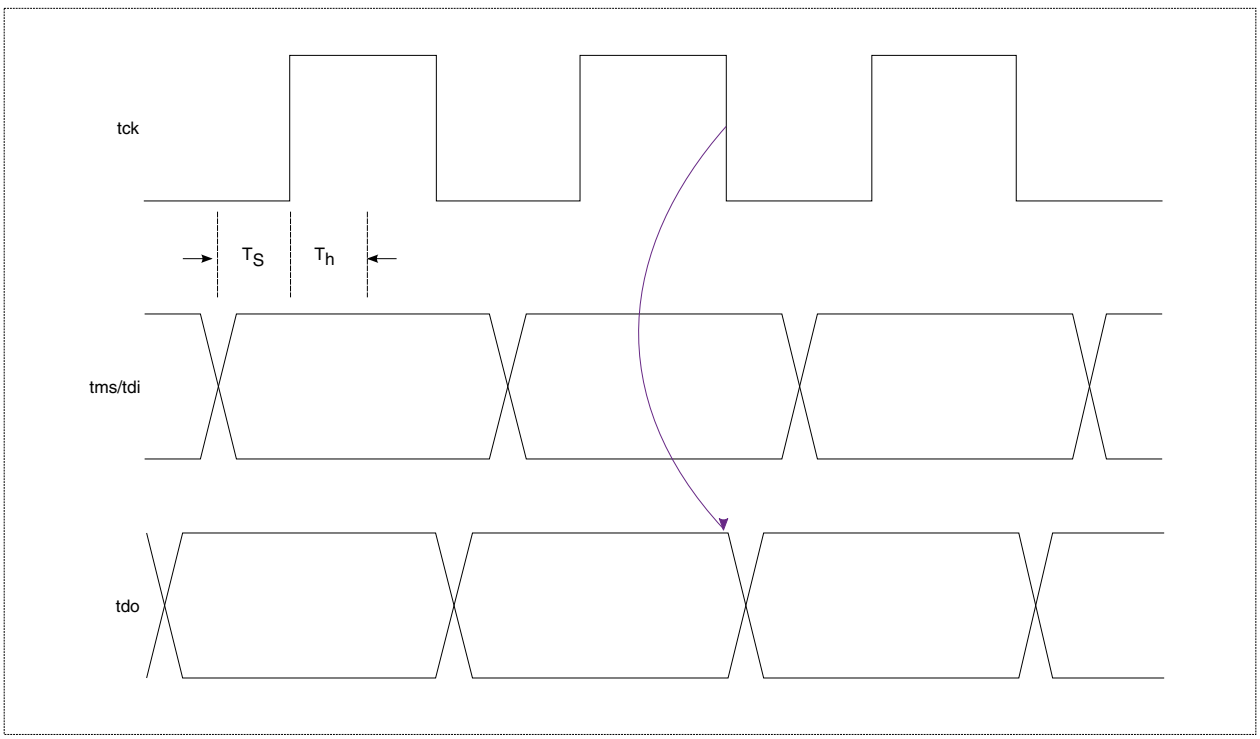


Figure 54-5. JTAG Interface Timing

NOTE

Setup (Ts) and hold (Th) times for tms and tdi are defined in the .lib files. The tdo signal transitions on the falling edge of tck.

The maximum frequency of operation for the JTAG port is 35 MHz. For detailed timing information, refer to the .lib files in the database deliverables.

If you are using the JTAG interface for off-chip connections, you must instantiate appropriate Low Voltage Transistor-Transistor Logic (LVTTL) input buffers in the ASIC for the JTAG inputs. To drive the tdo pin in compliance with the JTAG specification, you must also instantiate a tri-state output buffer.

54.5.2 Silicon Testing

This section describes SATA2 PHY interfaces and features that can be used in silicon testing.

54.5.2.1 Boundary Scan Port

The Boundary Scan port on the SATA2 PHY enables any ASIC using the macro to be fully compliant with the 1149.1-2001 and 1149.6-2003 standards.

The SATA2 PHY contains scannable, sequential elements to support ACJTAG and boundary scan. These elements are connected in a scan chain and made available on the ASIC interface. Concatenate this scan chain with any other scan chains on the ASIC.

54.5.2.1.1 Per-Lane Block Diagram

The figure below shows the Boundary Scan circuitry included in each lane. The signals to control the Boundary Scan circuitry are derived from the Scan Port interface pins. There are no boundary scan cells in the Clock module.

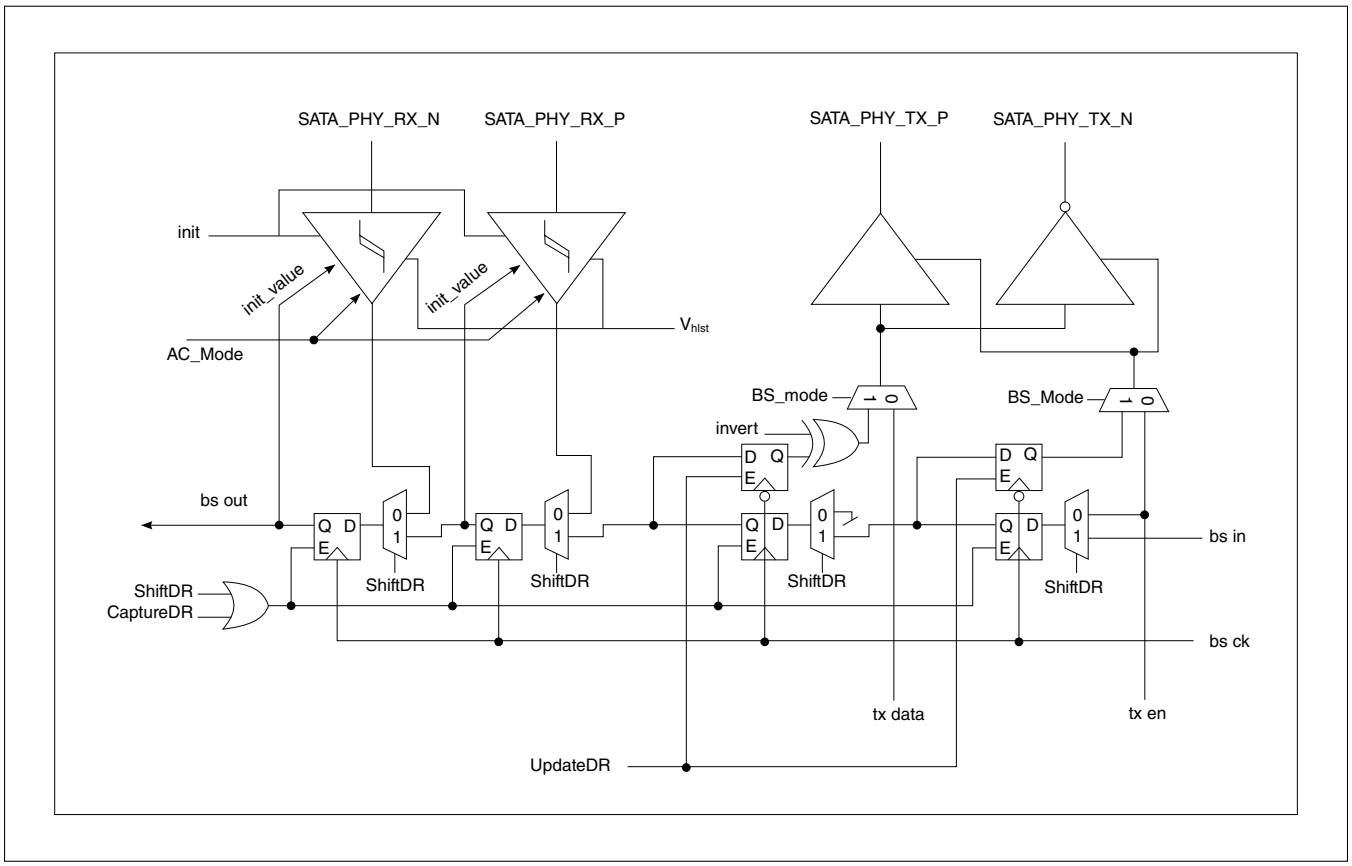


Figure 54-6. Per-Lane ACJTAG Details

The following figures show the function of the comparator associated with the receive pins shown in the previous figure.

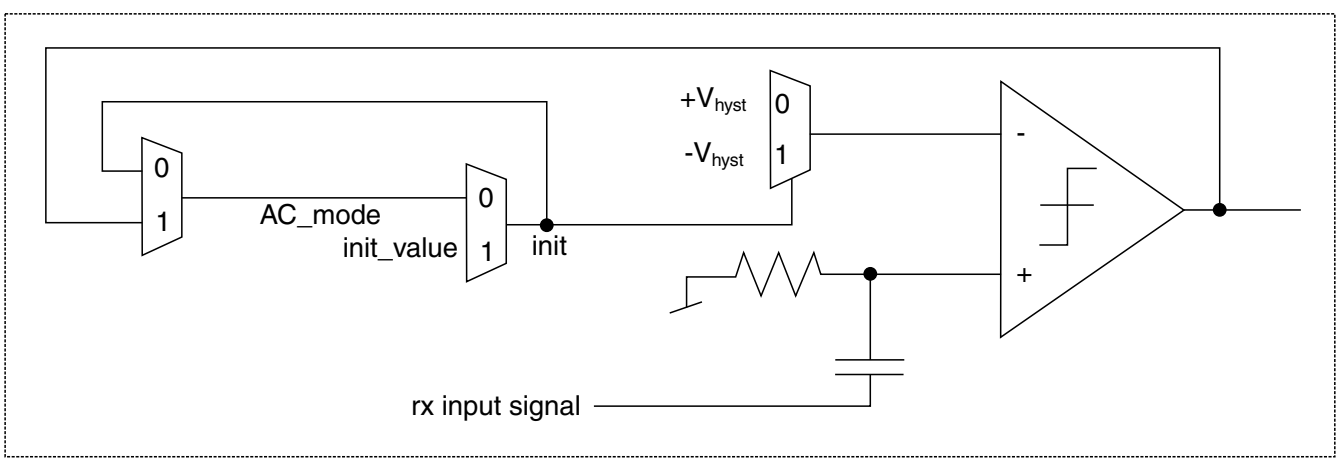


Figure 54-7. Comparator in AC mode

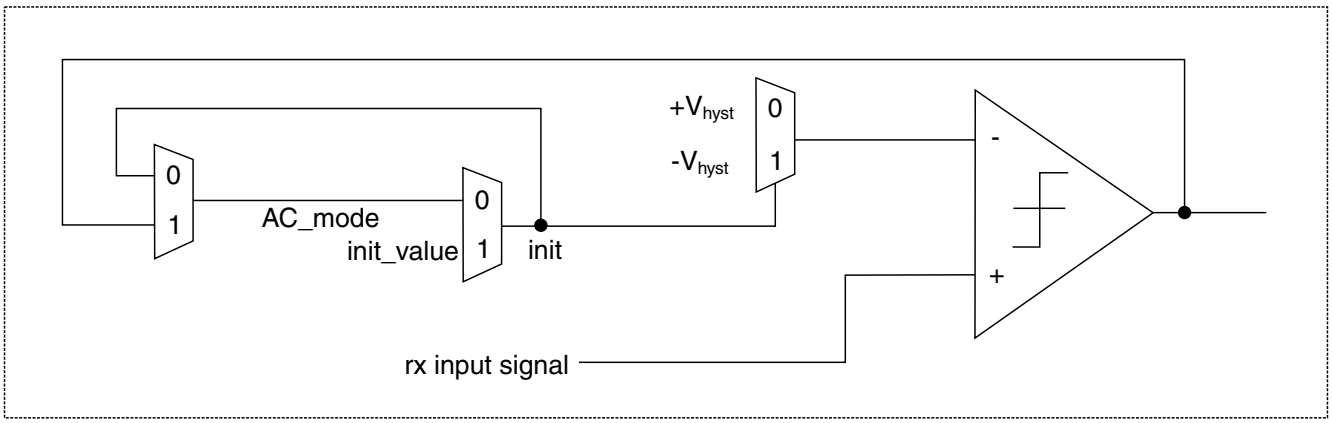


Figure 54-8. Comparator in DC mode

54.5.2.2 JTAG Interface Silicon Testing

The SATA2 PHY's JTAG interface provides access to an internal TAP controller implementing 1149.6-2003- compliant IDCODE and USERCODE instructions.

In addition, the CRSEL instruction is available for writing and reading registers in the SATA2 PHY. This instruction provides access to features available for bench characterization and ATE (manufacture) testing. Read and/or write register access for normal operations is neither necessary nor recommended.

54.5.2.2.1 Interface Options

The SATA2 PHY's JTAG interface excludes the optional TRSTb reset pin as defined in the 1149.6-2003 specification.

54.5.2.2.2 Resets

Entry into the TEST-LOGIC-RESET state clears the IR register and DR-shift register; no other registers are affected.

Note that due to the JTAG state machine's topology, TEST-LOGIC-RESET is entered regardless of the JTAG machine's current state-by clocking in at least five consecutive 1s on TMS.

After a power-on reset, TEST-LOGIC-RESET is the default state for the JTAG state machine.

54.5.2.2.3 IR Codes

The table below lists the supported JTAG instructions. The least-significant bit (LSB) of the IR and DR registers is the first bit shifted into TDI.

Table 54-22. Supported JTAG Instructions

Instruction	IR	DR	Operation
BYPASS	All others	1 bit	TDI to TDO bypass (bit is preloaded to 1)
IDCODE	8'h01	32 bits	Shift out ID Code
USERCODE	8'h0d	32 bits	Shift out USER Code
CRSEL	8'h31	18 bits	Control register access (see Control Register Operations)
DSCAN	8'h51	TBD bits	MUX-D scan testing
ATEOVRD	8'h5d	17 bits	Override reset, MPLL settings, and technology-specific parameters

54.5.2.2.4 ID Code

The table below provides the 32-bit JTAG ID code for the SATA2 PHY.

Table 54-23. JTAG ID for the SATA2 PHY

Bit Range	Signal	Value
31:16	ID_VAL_HI	1017
15:0	ID_VAL_LO	74CD

54.5.2.2.5 USER Code

The 32-bit JTAG USER code reflects the hard-wired programming of the SATA2 PHY in an ASIC. Various bits reflect the status of ASIC interface pins defined in Per-Transceiver Control and Status Signals. The table below provides the register mapping for the USER code.

Table 54-24. JTAG USERCODE

Bit Range	Signal
31:0	000, use_refclk_alt, mpll_prescale[1:0], mpll_ncy[4:0], mpll_ncy5[1:0], wide_xface, fast_tech, vp_is_1p2, vph_is_3p3, tx_lvl[4:0], los_lvl[4:0], acjt_lvl[4:0]

54.5.2.2.6 Control Register Operations

Internal registers can be accessed through the JTAG interface when operating with the CRSEL instruction (for a list of supported JTAG instructions, see Table 54-22).

The JTAG controller implements two shadow registers that access the macro's register space: an address register (Addr_reg) and data register (Data_reg).

Addr_reg is loaded with the address of the register to be accessed. Data_reg holds data to be written to or data that has been read from the addressed register. Four operations involving Addr_reg and Data_reg and the reading and writing of registers are performed with the shifting of 18 bits (16 bits of data or address and 2 bits of opcode) through the DR scan path of the JTAG state machine. The first 16 bits that are shifted out when in Shift-DR are the current contents of the Data_reg register for any operation (before the register read occurs).

Data_reg is also updated with the 16 bits of address or data shifted in as part of one of the following operations.

- Address (op = 2'b00): Addr_reg and Data_reg are loaded with the 16 bits of address shifted in.
- Write (op = 2'b01): Data_reg is loaded with the 16 bits of data shifted in. The register addressed by the contents of Addr_reg is written with this value.
- Read (op = 2'b11): Addr_reg is loaded with the 16 bits of address shifted in. A read of the register addressed by the contents of Addr_reg is initiated. The read data is latched into Data_reg with the passing through of the Capture-DR state. Therefore, a subsequent operation is required to shift out the read data.
- Data (op = 2'b10): This No Operation (NOP) operation can be used to shift out the current contents of Data_reg (for example, after a read operation). Data_reg is consequently loaded with the 16 bits shifted in.

NOTE

A standalone register read requires two passes down the DR scan path to actually do the read and retrieve the data. However, the second pass can be overlapped or effectively pipelined with a subsequent operation.

54.5.2.2.7 JTAG Override Register (jtag_ovrd)

Reset, MPLL controls, and technology-specific parameters can be overridden using the JTAG interface.

The Instruction register (ATEOVRD)-at JTAG IR address 8'h5d-enables the JTAG Override register to be written to

Reset Value: 17'b0 0000 0000 0000 0000

Table 54-25. JTAG Override Register (jtag_ovrd)

Field Name	Bits	Description
jtag_ovrd	0	JTAG override. Enables control of this register. Active high.
reset	1	Reset override. Active high.
mpll_ck_off	2	Reference clock stop enable override. Active high.
mpll_pwron	3	MPLL power-on override. Active high.
use_refclk_alt	4	Alternate reference clock control override. Active high.
mpll_prescale	6:5	Reference clock prescaler control override. For more information, see the mpll_prescale[1:0] signal description in Table 1.
mpll_ncy	11:7	MPLL x4 multiplier override. For more information, see the mpll_ncy[4:0] signal description on Table 1.
mpll_ncy5	13:12	MPLL x1 multiplier override. For more information, see the mpll_ncy5[1:0] signal description on Table 1.
fast_tech	14	Fast technology flag override. For more information, see the fast_tech signal description on Table 1.
vp_is_1p2	15	Low-voltage supply is 1.2-V override. For more information, see the vp_is_1p2 signal description on Table 1.
vph_is_3p3	16	High-voltage supply is 3.3-V override. For more information, see the vph_is_3p3 signal description on Table 1.

54.5.2.3 Parallel CR Control Port Testing

ATE and bench characterization features are accessed through internal registers via the JTAG interface or Parallel CR Control port. Normal SATA2 PHY startup and operation do not require the use of neither the JTAG interface nor the Parallel CR Control port.

The Parallel CR Control port comprises two internal registers (Address and Data) as well as control signals to manipulate these registers and to use them to perform control register reads and writes. The cr_data_in signal can be used to write to either the Address or Data register (depending on the control signals). The cr_data_out signal is always the Data register's output.

Communicating with the control port itself is accomplished with a standard four-phase asynchronous handshake using the four CR request lines and the cr_ack signal.

This protocol can be summarized by the following sequence.

1. The ASIC waits until the cr_ack signal is deasserted from a previous operation. At this point, all request signals are deasserted.

2. The ASIC sets up data on cr_data_in (if necessary) and asserts the target request signal.

3. After capturing the data and completing the requested operation, the SATA2 PHY sets up data on cr_data_out (if necessary) and asserts the cr_ack signal.

4. After capturing data from cr_data_out (if necessary), the ASIC deasserts the request signal.

5. The SATA2 PHY deasserts the cr_ack signal to indicate that the SATA2 PHY is ready for the next request.

54.5.2.3.1 Addressing

Before reading or writing a register, an address must be supplied and latched into the port's Address register.

This transaction requires the following steps.

1. Supply the address on cr_data_in.

2. Assert the cr_cap_addr signal.

3. Wait for the cr_ack signal to be asserted.

4. Deassert cr_cap_addr.

5. Wait for cr_ack to be deasserted.

The following figure shows timing for this transaction.

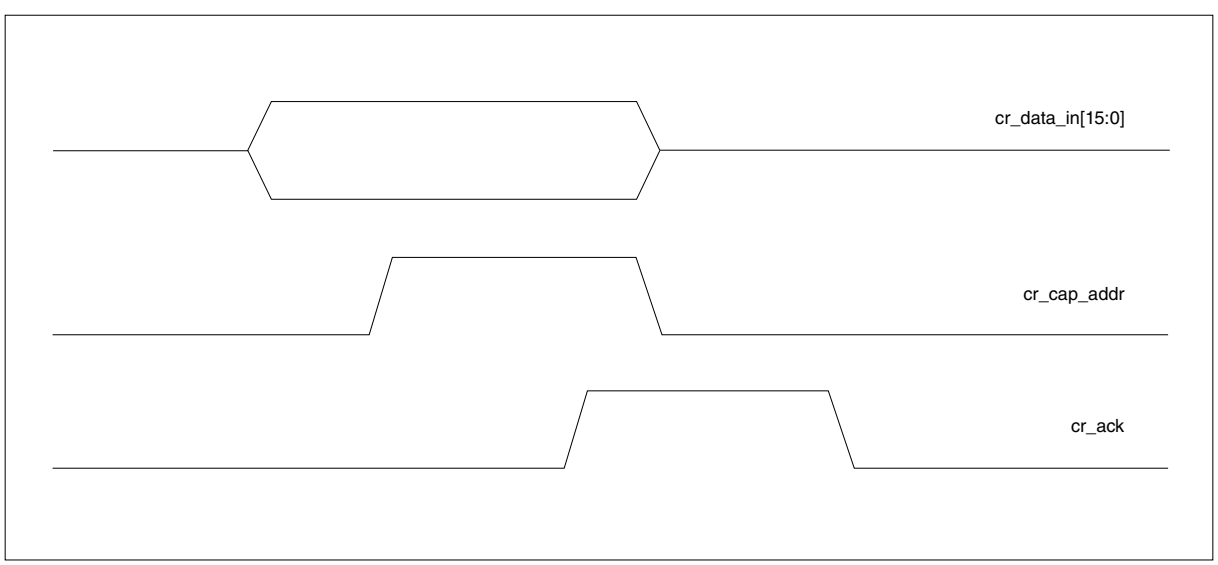


Figure 54-9. Capture Address Transaction (Precedes Read or Write)

54.5.2.3.2 Register Write

Writing to an internal register requires providing the write data and latching it into the port's Data register, then executing the write itself.

This transaction requires the following steps.

1. Supply the data on cr_data_in.
2. Assert the cr_cap_data signal.
3. Wait for the cr_ack signal to be asserted.
4. Deassert cr_cap_data.
5. Wait for cr_ack to be deasserted.
6. Assert the cr_write signal.
7. Wait for cr_ack to be asserted.
8. Deassert cr_write.
9. Wait for cr_ack to be deasserted.

The following figure shows timing for this transaction.

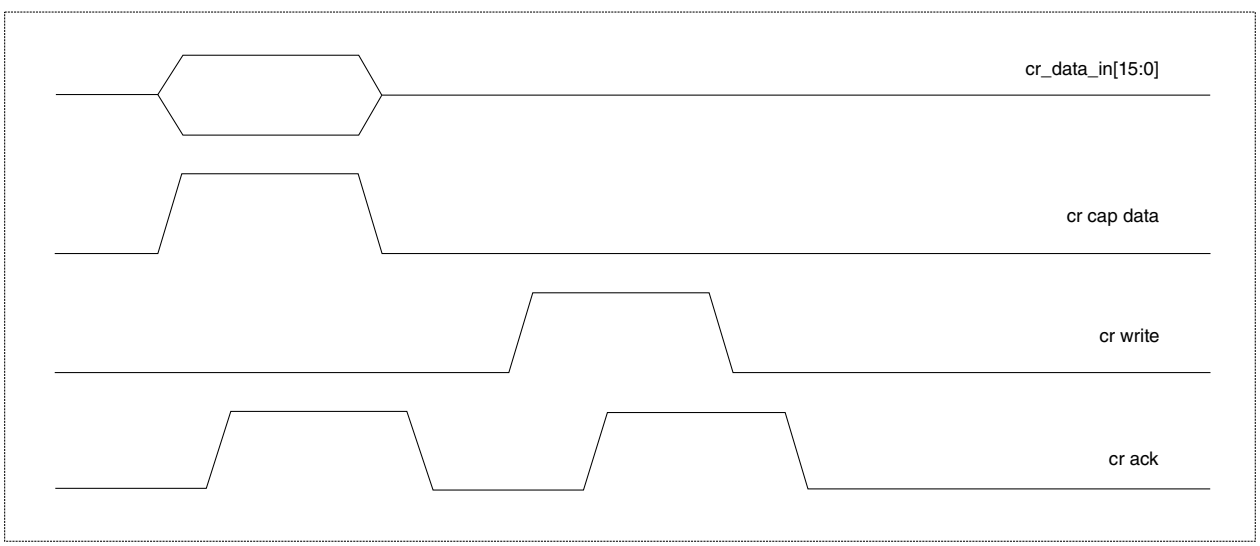


Figure 54-10. Write Transaction

54.5.2.3.3 Register Read

Reading from an internal register requires the following steps (to latch the read value into the port's Data register).

1. Assert the cr_read signal.
2. Wait for the cr_ack signal to be asserted.
3. Capture the data from cr_data_out[].
4. Deassert cr_read.
5. Wait for cr_ack to be deasserted.

The figure below shows timing for this transaction.

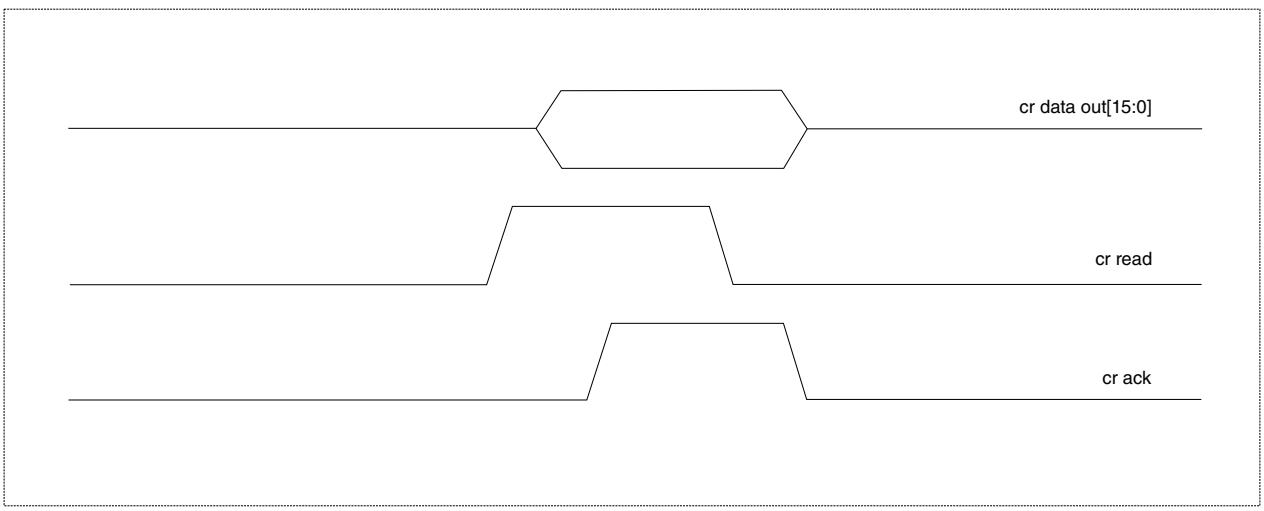


Figure 54-11. Read Transaction

54.5.2.4 Diagnostic Features

The SATA2 PHY provides the following diagnostic features to enhance bench characterization and ATE testing.

- Loopback functions:
  - Digital serial loopback
  - Serial loopback for wafer probe only

    - For information about the loopback functions, see Loopback Functions.
    - Asynchronous operation in a synchronous test environment for enhanced coverage of functional tests: For information about asynchronous operation, see Asynchronous Operation.
- BERT independent per lane:
    - 7th- and 15th-order polynomial pattern generation and recognition
    - Generation of simple test patterns
    - Byte error counting from polynomial pattern
    - For information about the BERT, see Byte Error Rate Tester.
- Margining:
    - Voltage margining with 10-bit resolution, synchronous or asynchronous operation
    - Phase margining with sub-ps resolution, synchronous or asynchronous (when number of lanes is greater than one)
    - For information about margining, see Margining.
- High-resolution scope per Rx signal pair:
    - Acquisition of eye or signal from 7th- or 15th-order polynomial patterns
    - For information about scope function, see Scope Function.
- Analog DC test:
    - Signal selection matrix embedded in the SATA2 PHY
    - 10-bit A/D converter
    - Selection and measurement of any individual Rx and Tx termination resistors
    - For information about the SATA2 PHY's analog DC test capabilities, see Analog DC Test Capabilities.
- Limit testing:
    - Specification of high/low limits
    - Determination of whether register read was within limits
    - Determination of whether difference between register read values was within limits
    - For information about limit testing, see Limit Testing.
- Integrated test modes:
    - Bypass test mode
    - Burn-in test mode
    - IDDQ test mode
    - For information about the SATA2 PHY's integrated test modes, see Integrated Test Modes.

54.5.2.4.1 Loopback Functions

The figure below depicts the SATA2 PHY's loopback functions.

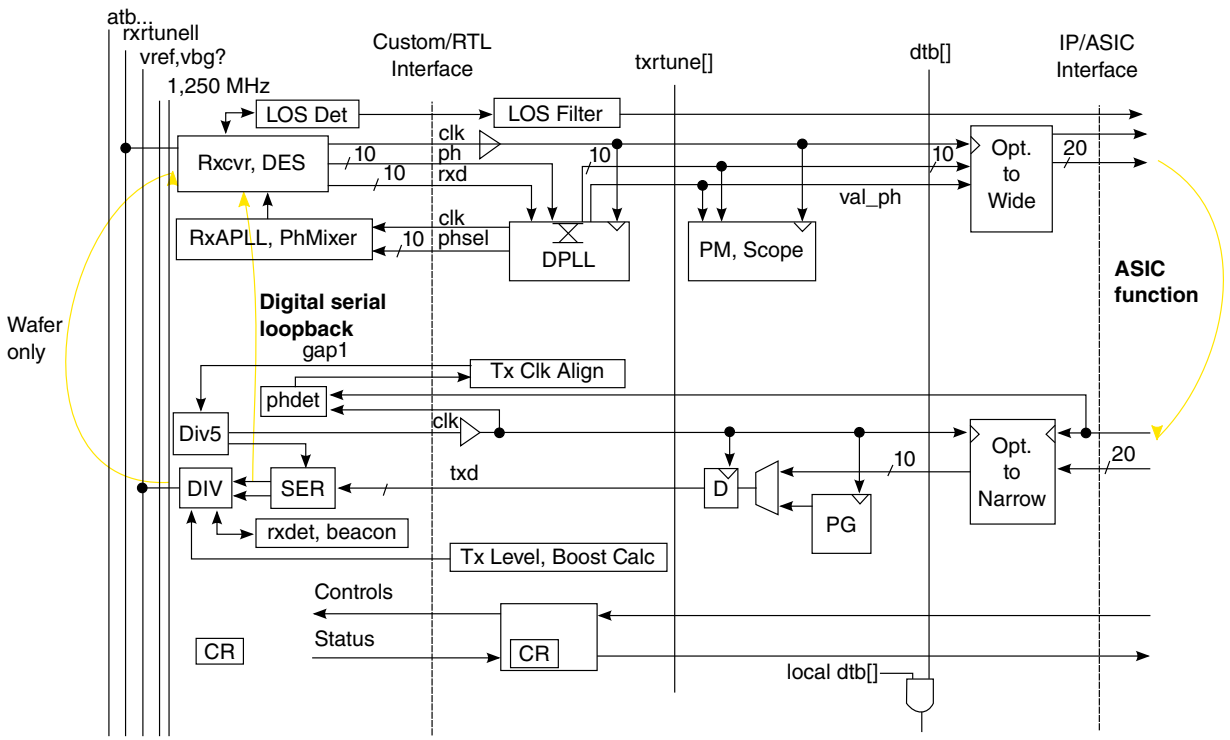


Figure 54-12. Loopback Functional Diagram

54.5.2.4.1.1 Rx-to-Tx Parallel Data Loopback

This loopback function must be provided by the ASIC.

54.5.2.4.1.2 Tx-to-Rx Digital Serial Data Loopback

This loopback function selects serial data located just before the Tx driver and inserts the data into the Rx just after the first stage of the receiver.

This function enables verification of the entire datapath, except for actual output drivers and the Rx's first stage. This loopback function is useful for testing the ASIC and is activated by asserting the rxlbi_en field in the Receive Analog Control register (lanej.rx_ana.ctrl), as indicated in Receive Analog Control Register (lane0_RX_ANA_CONTROL).

54.5.2.4.1.3 Tx-to-Rx Serial Analog Loopback

This loopback function is intended to be used only in manufacturing test at wafer probe. Without wafer probes landing on the Tx or Rx serial differential pads, this loopback function tests the entire transceiver at wafer test, where providing loopback traces with sufficient signal integrity through a probe card is typically impossible.

This loopback function is activated by asserting the rxlbe_en field in the Receive Analog Control register (lanej.rx_ana.ctrl.rxlbe_en), as indicated in Receive Analog Control Register (lane0_RX_ANA_CONTROL).

Using this loopback function with typical packaged parts (even without connections to the Tx and Rx serial differential pins) would provide too much parasitic capacitance loading to enable operation at speed.

54.5.2.4.1.4 Full Analog Loopback for In-Package ATE Test

This loopback function requires the use of loopback traces in the ATE load board.

54.5.2.4.2 Asynchronous Operation

ATE test environments are typically synchronous, where only a single reference clock is available.

In addition, when loopbacks (internal to the PHY or wired into a package load board) are used in production test, only a single Clock module is typically part of the test setup; therefore, only synchronous operation is possible. However, a variety of circuit defects can exist that might have little or no effect on CDR operation when operating synchronously.

The SATA2 PHY addresses this coverage gap for multi-lane PHYs. Each pair of lanes is interconnected with the ability to use the recovered clock of the other member of a pair as reference for receive CDR.

One member (master) of the pair of lanes is programmed to ignore the received signal and produce a specified frequency offset by a specified number of ppm from the actual reference clock. The other member (slave) of the pair uses the synthesized offset clock as reference.

While receiving data synchronous to the actual reference clock, the slave's CDR must constantly slew to maintain a fixed recovered clock phase despite the offset frequency of the slave's reference. This task exercises the slave's DPLL and phase mixer in the same way they would operate during normal, asynchronous operation. Margining tests performed under this condition reveal any defects in the slave's CDR.

54.5.2.4.3 Byte Error Rate Tester

The Byte Error Rate Tester (BERT) comprises two independently programmed modules: a pattern generator and a pattern matcher/error counter.

54.5.2.4.3.1 BERT Pattern Generator

The following table provides a list of patterns that can be generated with the built-in pattern generator.

Table 54-26. Pattern Generator mode[2:0] Control

mode[2:0]	Description
000	Pattern generator is disabled.
001	15th order polynomial: $X^{15} + X^{14} + 1$
010	7th order polynomial: $X^7 + X^6 + 1$
100	Fixed 8- or 10-bit pattern from bottom of PAT0 field
101	2-byte DC balanced pattern constructed as {PAT0, ~PAT0}
111	4-byte DC balanced pattern constructed as {0x000, PAT0, 0x3FF, ~PAT0}

54.5.2.4.3.2 BERT Pattern Matcher and Error Counter

The pattern matcher is capable of synchronizing to and detecting errored bytes in multiple types of patterns. Errored bytes are counted in the error counter.

Note that there is no dependence on the pattern generator in the same lane. That pattern generator does not need to be enabled nor programmed for the same pattern.

The following table describes the mode[2:0] field, which selects the expected pattern and operating mode.

Table 54-27. mode[2:0] Control of the Pattern Matcher

mode[1:0]	Description
000	Disabled
001	lfsr15
010	lfsr7
011	d[n] = d[n-10]
100	d[n] = !d[n-10]
101	Reserved
110	Reserved
111	Reserved

NOTE

When using either the LFSR7 or LFSR15 pattern, the rx_align_en needs to be de-asserted at the pin or through the register override.

For modes 1 and 2, the pattern matcher operates by generating the expected pattern and synchronizing the generated pattern to the incoming pattern. Synchronization and error counting are initiated by asserting and deasserting the sync bit. Therefore, during synchronization in high-error-rate conditions, synchronization can fail if an error occurs during the last 15 bits prior to clearing of the sync bit.

The error counter is 22 bits wide.

The contents are presented according to these rules:

- When the contents are less than 2^{15}, the bottom 15 bits are presented in the count field and the ov14 field is cleared.
- When the contents are greater than 2^{22} but less than or equal to 2^{15}, the top 15 bits are presented in the count field and the ov14 field is set. This result effectively scales the contents down by a factor of 2^{7}.
- When the 22-bit counter overflows, the count field is set to 0x7FFF, and the ov14 field is set.

The error counter is typically used in a "clear after read" mode. The error registers are reset on a read-only when the pattern matcher is enabled. If the pattern matcher is disabled, the registers return the error count that was indicated when the pattern matcher was disabled and never reset the error counter.

54.5.2.4.4 Margining

Margining is the process of learning about the quality of a communications system by studying how its BER is changed when known impairments are introduced to the system.

Typical BER targets for systems are extremely low (for example, 1e-18; at 2.5 Gbps; that is, one error every 12.7 years). Margining makes it possible to extrapolate from a short set of measurements to project bounds on the actual error rate, assuming there are no additional sources of low-rate errors (for example, metastability), which do not occur during the measurement, but would dominate the BER over large periods.

One common margining technique-phase margining-is referred to as a "bathtub curve." The figure below shows an example of this technique, where the extrapolated phase margin is a measurement of margin between the extrapolated system performance and the performance target.

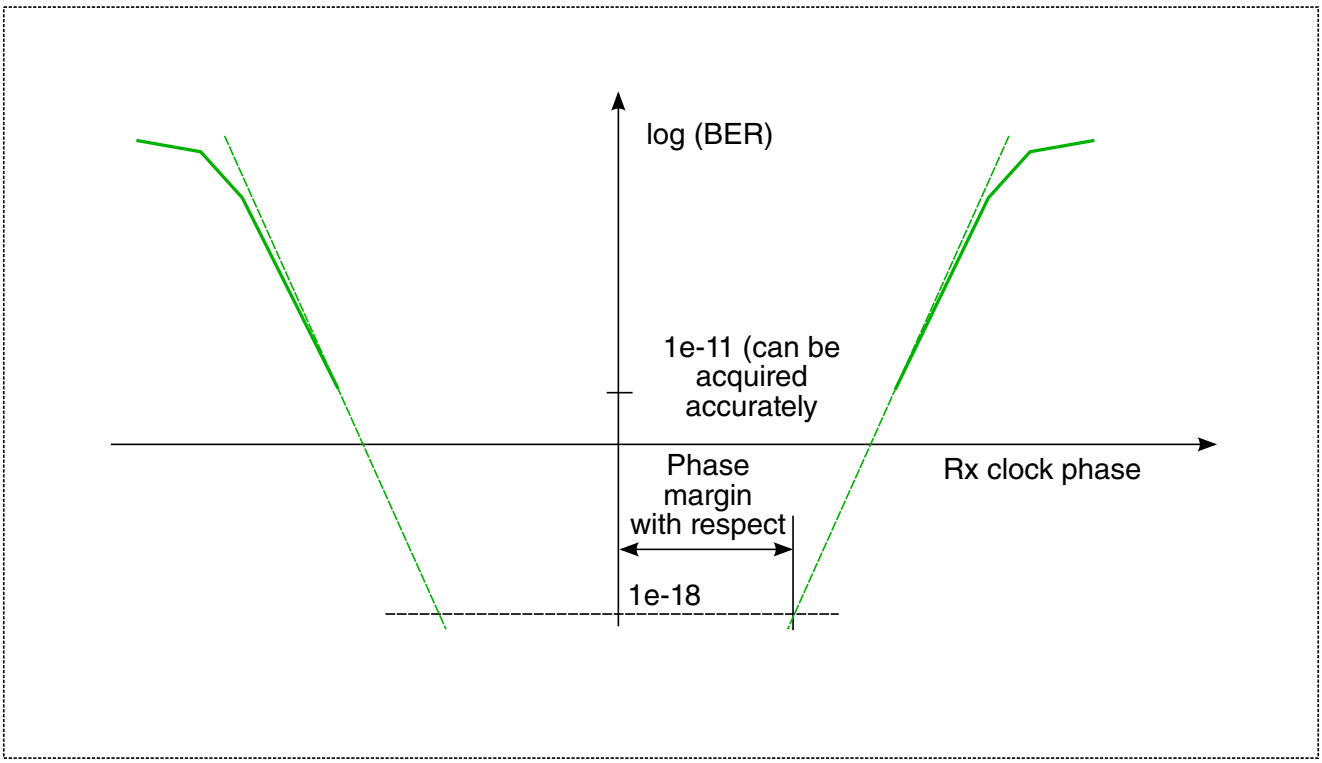


Figure 54-13. Example of Phase Margining Technique

Margining can be used on the testbench to study the effects of other changes (for example, crosstalk, trace lengths, power supply noise, temperature, power supply voltage) on system performance.

In ATE testing, during test time intervals in the order of 100 ms, simple error counting can pass defective parts that might produce BER results in the order of 1e-8. However, during that same period, margining can be used to provide a better bound on the performance of shipped parts.

54.5.2.4.4.1 Phase Margining

Phase margining is more common than voltage margining, but phase margining requires you to turn off (or freeze) the receiver's clock recovery-an important receiver function. Generally, to perform phase margining, you must ensure that the transmitter and receiver are operating from a common clock.

Synchronous phase margining requires the following steps.

1. 1. Provide a common clock to the transmitting and receiving devices.

2. Determine the mean selected phase of the receiver's CDR by reading the DPLL Phase register (lane0_DPLL_PHASE) of the lane to be margined-a number of times and averaging the values.

3. Enable the pattern generator in the relevant transmitter. For information about the BERT pattern generator, see BERT Pattern Generator.

4. Enable the pattern matcher and error counter in the receiver under test. For information about the BERT pattern matcher and error counter, seeBERT Pattern Matcher and Error Counter.

5. Freeze the receiver's CDR.

In lanej.rx_ctl, set the phug_value field to 1'b1, the ovrd_dpll_gain field to 1'b1, and the frug_value field to 2'b00 (default).

Set lanej.freq to 13'b0_0000_0000_0000.

In lanej.rx_ctl, set the phug_value field to 2'b00.

6. Force a selected phase offset from the mean phase identified in step 2 by adding a value to the value measured in step 2 and writing the sum to lanej.phase.

7. Measure the BER at this phase offset.

8. Repeat steps 6 and 7 to get the BER versus programmed phase offset.

5.2.5.4.2 Voltage Margining

Voltage margining can be performed in an asynchronous environment without shutting down the CDR, because the receiver is clock-forwarded. Voltage margining is performed by adding an offset voltage to the received signal.

Note that when the applied offset voltage is large relative to the peak-to-peak received signal, the Rx CDR's phase detector develops a dead zone that adds to the sampling jitter, causing additional loss in margin.

Voltage margining requires the following steps.

1. Enable the pattern generator in the relevant transmitter. For information about the BERT pattern generator, see BERT Pattern Generator.

2. Enable the pattern matcher and error counter in the receiver under test. For information about the BERT pattern matcher and error counter, see BERT Pattern Matcher and Error Counter.

3. Enable the margining DAC in the Clock module.

4. Set the DAC to its midrange by setting clock.dac_ctl.dac_val to 10'b01_1111_1111 (511). The DAC is 10 bits wide and the DAC's midrange corresponds to a zero offset.

5. Enable the DAC by setting clock.dac_ctl.dac_mode to 2'b11.

6. Program an additive offset voltage by changing the value of clock.dac_ctl.dac_val from 511. The resolution (1 LSB) of the DAC is VP25 x 279e-6 volts.

7. Measure the BER at this voltage offset.

8. Repeat steps 4 and 5 to get the BER versus programmed voltage offset.

54.5.2.4.5 Scope Function

Scope function (capturing and viewing the signals received at each input) can be achieved by combining the SATA2 PHY's signal acquisition capabilities with software running on an external host and utilizing external graphics display capabilities.

Note that scope function differs from the function of expensive lab equipment in some important ways; for example:

- Scope: The actual signal delivered to the receiver's slicers is measured. When a lab hardware scope is used, different reflections are present. In some cases, an alternative board-not the board with the receiver-is plugged in, enabling use of an external connector. Consequently, the reflection environment is changed entirely. In other cases, an active probe is used to pick up the signal near the receiver. This technique picks up reflections from the receiver that are not actually present at the receiver (at least 11-20 mm from the closest possible pickup point).
- Scope: The signal is acquired with the actual bandwidth limitations of the slicers in the receivers. This situation might not suit your measurement goals.
- A limitation of the on-die scope is that only periodic signals of known periodicity can be acquired, though the periodicity can be as long as 2^16 bytes.
- Scope: Jitter of the receiving device's Rx Analog Phase-Locked Loop (APLL) and MPLL is present in the measurement. This jitter is generally larger than the sampling jitter of a quality piece of bench equipment. In a loopback situation (same MPLL for both Tx and Rx), much of this jitter is cancelled, yielding the appearance of Tx jitter that is lower than is actually present.
- Scope function is analogous to an undersampling scope, not a real-time, oversampling scope.
- Bench hardware typically has voltage and time accuracy that are carefully specified and directly traceable to external standards. Tracing voltage, offset, and phase accuracy to external standards is more indirect.

The scope's greatest strength is its ease of use in debugging and understanding a system in situ. The scope is not intended to replace traditional lab equipment for the purpose of compliance testing, but the scope provides a nondestructive view of incoming data.

54.5.2.4.6 Analog DC Test Capabilities

This section describes the SATA2 PHY's analog DC test capabilities.

54.5.2.4.6.1 Analog Test Bus

The analog test bus (ATB) comprises four signals routed throughout the Active section, through the Clock module and each lane.

Some of these signals can be brought to the internal ADC while others can be brought to any SATA_TX or SATA_RX pin. The table below describes the ATB signals.

Table 54-28. Analog Test Bus Signals

Signal	Description	Can be Brought to:
atb_s_p	Zero current "sense" connections	ADC, SATA_TXP, SATA_RXP
atb_s_m	Zero current "sense" connections	ADC, SATA_TXM, SATA_RXM
atb_f_p	Large current (up to about 3 mA) "force" connection	SATA_TXM, SATA_TXP, SATA_RXM, SATA_RXP

Many internal circuits have connections that can be enabled to these buses. Note that these connections are intended to be used in a very-low-frequency (near DC) situation. Series resistance between connections on the bus are typically > 30 ohms, while bus parasitic capacitance can be multiple picoFarads, yielding bandwidths below 2 MHz. When connections are made to nodes carrying high-frequency signals such as SATA_TX or SATA_RX serial pins, explicit 10-k ohm resistors are used so that connecting the ATB bus does not impair the lane's operation.

The figure below shows a high-level map of the ATB.

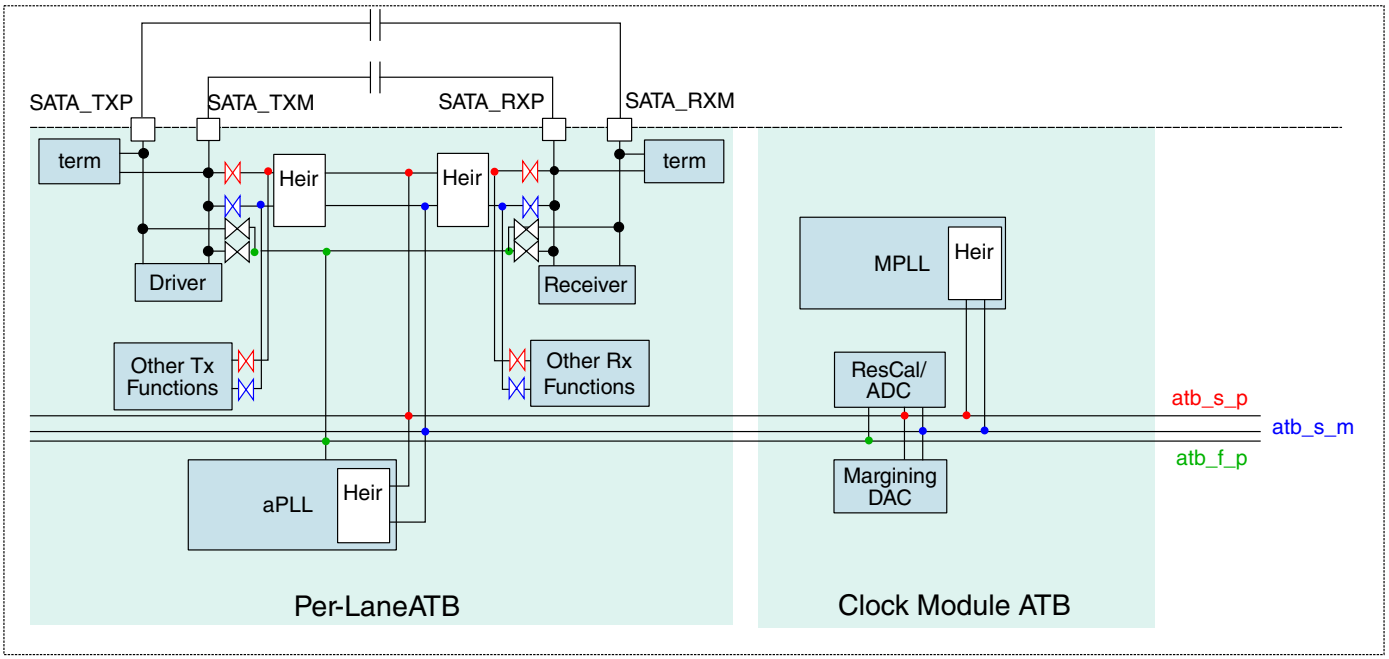


Figure 54-14. High-Level Map of ATB

Bringing a selected internal signal to either a package pin or the internal ADC is a matter of activating the correct combination of hierarchical switches and leaf-level switches. Control of these switches is distributed across several registers, as described in the table below.

Table 54-29. Location of ATB Switch Controls

Circuit Area	Control Register
Transmit	Hierarchical switch: "atb_en" in Receive Analog Control Register (lane0_RX_ANA_CONTROL) Leaf-level switches: Transmit ATB 1 Control Register (lane0_TX_ANA_ATBSEL1) and Transmit ATB 2 Control Register (lane0_TX_ANA_ATBSEL2)
Receive	Hierarchical switch: "atb_en" in Receive Analog Control Register (lane0_RX_ANA_CONTROL) Leaf-level switches: Receive ATB Register (lane0_RX_ANA_ATB)
CDR aPLL	Hierarchical switch: "atb_sense_sel" in Rx PLL Programming 2 Register (lane0_PLL_PRG2) Leaf-level switches: Rx PLL Measurement Register (lane0_PLL_PRG3)
MPLL	Hierarchical switch: "atb_sense_sel" in Rx PLL Programming 2 Register (lane0_PLL_PRG2) Leaf-level switches: MPLL Test Register (clock_MPLL_TEST)

Disabling Rx Termination

To bring ATB signals out on the SATA_RX pins, it is recommended that you disable the resistive terminations that are otherwise present at these pins.

When using the SATA_RX pins connected to an external measuring device, disable the receive termination resistors by deasserting the rx_term_en bit. In addition, disable the LOS by setting the los_ctl bits to 2'b00.

The SATA_TX pins must not be used for this purpose, because the driver contains some impedances and leakage that cannot be completely eliminated.

54.5.2.4.6.2 10-Bit DAC

The 10-bit DAC is required for voltage margining and is used as part of the 10-bit AD converter (see 10-Bit ADC).

In addition, the DAC can be connected to the DC test bus for other purposes and is generally controlled by the DAC Control register (clock.dac_ctl). (For information about this register, see "DAC Control Register (clock.dac_ctl)" on page 47.)

To use the 10-bit DAC:

1. Place the DAC in one of the DAC operating modes by setting clock.dac_ctl.dac_mode to a value of 4-7 (see DAC Control Register (clock_DAC_CTL)).
2. Connect the DAC output to the global atb_s_p bus by setting clock.rtune_ctl.dac_chop (see Resistor Tuning Control Register (clock_RTUNE_CTL)).
3. To produce the intended output voltage, write a value to clock.dac_ctl.dac_val (see DAC Control Register (clock_DAC_CTL)).

54.5.2.4.6.3 10-Bit ADC

The 10-bit ADC uses circuit components that are present in the Clock module, because the 10-bit DAC is required for voltage margining, and the offset calibrated comparator is present for resistor calibration.

54.5.2.4.7 Limit Testing

In most ASIC production test environments, the tests simply run a vector (drive controls into the DUT and verify that the correct output is produced) due to either limitations of the tester itself or time constraints in the development of more sophisticated tests. Therefore, tasks that would otherwise seem simple (for example, reading a register and verifying that the result is < 20) become effectively impossible.

The limit testing feature can be used to place upper and lower bounds on a masked version of a register read or to place bounds on a masked version of the difference between two masked register reads.

The figure below shows the hardware functions involved in limit testing.

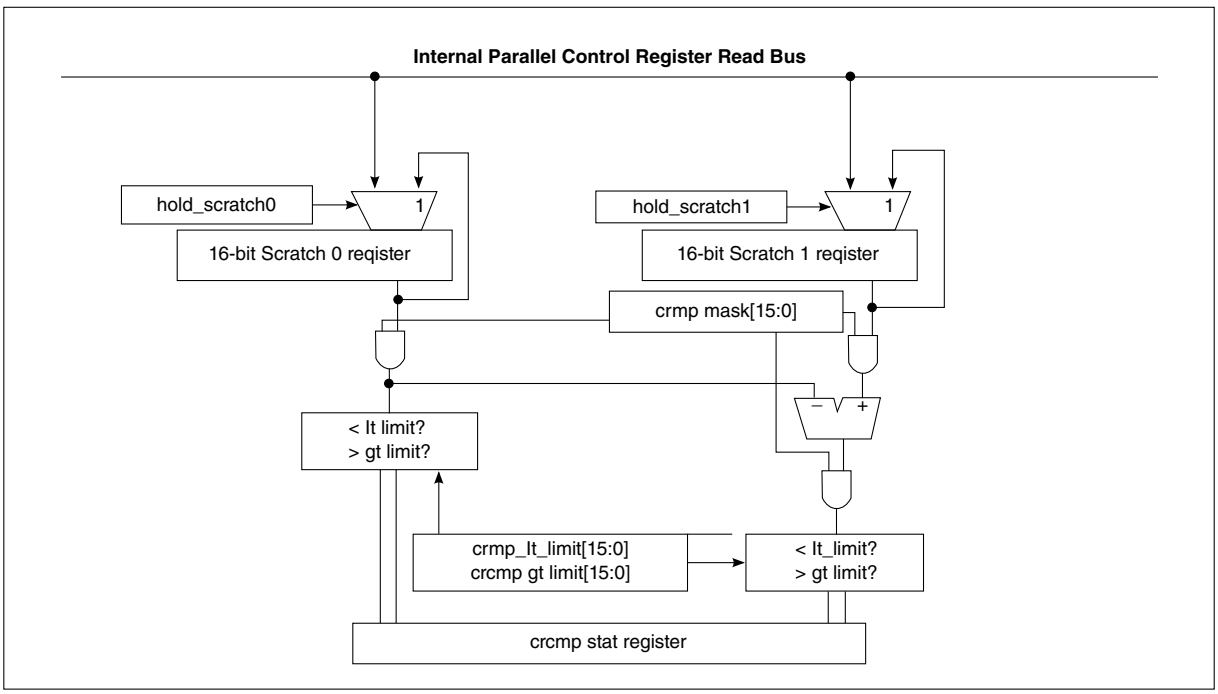


Figure 54-15. Hardware for Limit Testing

Each internal scratch register is updated each time a register read occurs through the JTAG interface, unless the associated hold_scratch1 or hold_scratch0 bit is set.

Note that the values written to the crcmp_lt_limit and crmp_gt_limit registers are also masked with the crcmp_mask field before being used.

54.5.2.4.8 Integrated Test Modes

The SATA2 PHY integrates the following test modes, which can be used to enhance characterization and ATE testing.

54.5.2.4.8.1 IDDQ Test Mode

To enable IDDQ test mode, set the pddq_h signal to 1'b1 (I/0 voltage level). This setting powers down circuitry that cannot be powered down as described in Power-Down Sequences.

All preconditioning of the SATA2 PHY must be done before asserting pddq_h, because the device becomes non-functional when pddq_h is set to 1'b1.

54.5.2.4.8.2 Bypass Test Mode

Setting the test_byp_mode signal to 1'b1 connects the parallel data inputs to the parallel data outputs through a purely combinatorial path. This mode enables testing of the ASIC/ SATA2 PHY interface; the passthru.v Verilog model represents this mode.

This interface is designed to be orders of magnitude slower than the operational frequency, so no timing information is provided. However, if necessary, using 1ns for a hold time would be a very conservative number.

54.5.2.4.8.3 Burn-In Test Mode

Setting the test_burnin_mode signal to 1'b1 enables the SATA2 PHY for burn-in testing. An on-board oscillator is activated and multiplexed to the prescaler inputs to be used as the reference clock.

This oscillator removes the requirement for an external reference. As much circuitry as possible is toggled, and unused circuitry is biased so that devices do not degrade asymmetrically.

54.5.2.4.9 Burn-In Test Requirements

For burn-in testing, the IP must be placed in a mode so that the IP's circuitry is stressed in its intended mode of operation.

This requirement involves providing the IP a reference clock, applying power, powering up the device as described in Power-Up Sequences, and sending parallel data to the transmitter. Externally, the transmitter pins must be AC-coupled through a capacitor (typically 0.1 µF) to the receiver, and the rbias resistor must be connected. Burn-in test mode satisfies this requirement-power must be applied to the IP and Burn-in test mode must be activated. Externally, the rbias resistor must be connected.

54.5.2.5 ATE Testing

The features described in Diagnostic Features were used to develop an analog, production-ready test program. This test suite tests the analog functions of the IP using simple pass/fail vectors that can be used on a low-cost digital tester.

Test vectors are entered into the core through the JTAG interface. By request, a program for generating digital test patterns as well as documentation that provides implementation guidelines will be provided .

54.6 clock Memory Map/Register Definition

This section describes registers in the Clock module. Only one instance of each register exists in the Clock module, regardless of the number of lanes.

Registers in the Clock module have 16-bit addresses. These addresses are noted in the register map and in the section for the applicable register.

NOTE

SATA PHY registers are only accessible by the corresponding controller (SATA_P0_PCSR_PHYCR and SATA_P0_PCSR_PHYSR) or in debug through the JTAG port. SATA PHY is not memory mapped to processor address space, so the absolute addresses shown is the relative address and is not valid. See SATA Memory Map for more information.

clock memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1	Creg Compare Upper Limit Register (clock_CRCMP_LT_LIMIT)	16	R/W	0000h	54.6.1/4650
2	Creg Compare Lower Limit Register (clock_CRCMP_GT_LIMIT)	16	R/W	FFFFh	54.6.2/4651
3	Creg Compare Mask Register (clock_CRCMP_MASK)	16	R/W	FFFFh	54.6.3/4651
4	Creg Compare Control Register (clock_CRCMP_CTL)	16	R/W	0000h	54.6.4/4651
5	Creg Compare Status Register (clock_CRCMP_STAT)	16	R	0000h	54.6.5/4652
6	Scope Sample Count Register (clock_SCOPE_SAMPLES)	16	R/W	0100h	54.6.6/4653
7	Scope Count Result Register (clock_SCOPE_COUNT)	16	R	0000h	54.6.7/4653
8	DAC Control Register (clock_DAC_CTL)	16	R/W	01FFh	54.6.8/4654
9	Resistor Tuning Control Register (clock_RTUNE_CTL)	16	R/W	0020h	54.6.9/4655
A	ADC Output Register (clock_ADC_OUT)	16	R	0000h	54.6.10/ 4656
B	Spread Spectrum Phase Register (clock_SS_PHASE)	16	R/W	0000h	54.6.11/ 4657
C	JTAG Chip ID (High Bits) Register (clock_CHIP_ID_HI)	16	R	0011h	54.6.12/ 4657
D	JTAG Chip ID (Low Bits) Register (clock_CHIP_ID_LOW)	16	R	74CDh	54.6.13/ 4658
E	Frequency Status Register (clock_FREQ_STAT)	16	R	0000h	54.6.14/ 4658
F	Control Status Register (clock_CTL_STAT)	16	R	0000h	54.6.15/ 4659

Table continues on the next page...

clock memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10	Level Status Register (clock_LVL-STAT)	16	R	0000h	54.6.16/ 4661
11	Creg Status Register (clock_CREG_STAT)	16	R	0000h	54.6.17/ 4661
12	Frequency Override Register (clock_FREW_OVRD)	16	R/W	4547h	54.6.18/ 4662
13	Control Override Register (clock_CTL_OVRD)	16	R/W	0854h	54.6.19/ 4663
14	Level Override Register (clock_LVL_OVRD)	16	R/W	4210h	54.6.20/ 4664
15	Creg Override Register (clock_CREG_OVRD)	16	R/W	0040h	54.6.21/ 4665
16	MPLL Control Register (clock_MPLL_CTL)	16	R/W	0000h	54.6.22/ 4665
17	MPLL Test Register (clock_MPLL_TEST)	16	R/W	0000h	54.6.23/ 4667
18	Spread Spectrum Frequency Register (clock_SS_FREQ)	16	R/W	332Fh	54.6.24/ 4668
19	Clock Select Status Register (clock_SEL_STAT)	16	R	0000h	54.6.25/ 4669
1A	Clock Select Override Register (clock_SEL_OVRD)	16	R/W	0000h	54.6.26/ 4669
7F3F	Reset Register (clock_RESET)	16	W	0000h	54.6.27/ 4670

54.6.1 Creg Compare Upper Limit Register (clock_CRCMP_LT_LIMIT)

Address: 0x0001

Reset value: 16'b 0000 0000 0000 0000

This register contains the less-than-limit compare point.

Address: 0h base + 1h offset = 1h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	crcmp_lt_limit															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

clock_CRCMP_LT_LIMIT field descriptions

Field	Description
crcmp_lt_limit	Less-than-limit compare point

54.6.2 Creg Compare Lower Limit Register (clock_CRCMP_GT_LIMIT)

Address: 0x0002

Reset value: 16'b 1111 1111 1111 1111

This register contains the greater-than-limit compare point.

Address: 0h base + 2h offset = 2h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	crcmp_gt_limit															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

clock_CRCMP_GT_LIMIT field descriptions

Field	Description
crcmp_gt_limit	Greater-than-limit compare point

54.6.3 Creg Compare Mask Register (clock_CRCMP_MASK)

Address: 0x0003

Reset value: 16'b 1111 1111 1111 1111

This register contains the compare/scratch value mask.

Address: 0h base + 3h offset = 3h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	crcmp_mask															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

clock_CRCMP_MASK field descriptions

Field	Description
crcmp_mask	Mask for comparisons

54.6.4 Creg Compare Control Register (clock_CRCMP_CTL)

Address: 0x0004

Reset value: 16'b 0000 0000 0000 0000

This register contains the scratch space control bits.

Address: 0h base + 4h offset = 4h

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	Reserved						hold_ scratch1	hold_ scratch0
Reset	0	0	0	0	0	0	0	0

clock_CRCMP_CTL field descriptions

Field	Description
15–2 -	This field is reserved. Reserved
1 hold_scratch1	Scratch1 is not updated on register reads.
0 hold_scratch0	Scratch0 is not updated on register reads.

54.6.5 Creg Compare Status Register (clock_CRCMP_STAT)

Address: 0x0005

Reset value: 16'b xxxx xxxx xxxx xxxx

This register contains the results of scratch register comparisons to various limits.

Address: 0h base + 5h offset = 5h

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	Reserved		s1_s0_ outside	s0_outside	s1_s0_high	s1_s0_low	s0_high	s0_low
Write								
Reset	0	0	0	0	0	0	0	0

clock_CRCMP_STAT field descriptions

Field	Description
15–6 -	This field is reserved. Reserved
5 s1_s0_outside	Logical OR of S1_S0_LOW and S1_S0_HIGH Useful for determining if the difference is near signed zero.
4 s0_outside	Logical OR of S0_LOW and S0_HIGH Useful for determining if the value is near signed zero.
3 s1_s0_high	Masked (Scratch1 - Scratch0) is higher than CRCMP_HT_LIMIT.
2 s1_s0_low	Masked (Scratch1 - Scratch0) is lower than CRCMP_LT_LIMIT.
1 s0_high	Masked Scratch0 is higher than CRCMP_HT_LIMIT.
0 s0_low	Masked Scratch0 is lower than CRCMP_LT_LIMIT

54.6.6 Scope Sample Count Register (clock_SCOPE_SAMPLES)

Address: 0x0006

Reset value: 16'b 0000 0001 0000 0000

This register specifies the number of samples to count.

Address: 0h base + 6h offset = 6h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	scope_samples															
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

clock_SCOPE_SAMPLES field descriptions

Field	Description
scope_samples	The number of samples to count

54.6.7 Scope Count Result Register (clock_SCOPE_COUNT)

Address: 0x0007

Reset value: 16'b xxxx xxxx xxxx xxxx

This register provides the results of scope counting. A write to this register starts the counting process. A value of FFFF indicates that the count is still in progress.

Address: 0h base + 7h offset = 7h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	scope_count															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

clock_SCOPE_COUNT field descriptions

Field	Description
scope_count	Results of scope counting

54.6.8 DAC Control Register (clock_DAC_CTL)

Reset value: 16'b x000 0001 1111 1111

This register supports DAC values and controls.

Address: 0h base + 8h offset = 8h

Bit	15	14	13	12	11	10	9	8
Read	Reserved	dac_mode			ovrd_rtune_rx	ovrd_rtune_tx	dac_val	
Write								
Reset	0	0	0	0	0	0	0	1

Bit	7	6	5	4	3	2	1	0
Read	dac_val							
Write								
Reset	1	1	1	1	1	1	1	1

clock_DAC_CTL field descriptions

Field	Description
15 -	This field is reserved. Reserved
14–12 dac_mode	DAC output mode: 000 Powers down DAC 001 Reserved 010 High-range margining (VP25 x 418e-6 res) 011 Low-range margining (VP25 x 279e-6 res) 100 100% range DAC, 0% offset 101 36% range DAC, 0% offset

Table continues on the next page...

clock_DAC_CTL field descriptions (continued)

Field	Description
	110 36% range DAC, 33% offset 111 36% range DAC, 66% offset
11 ovrd_rtune_rx	Writes DAC_VAL[5:0] to the Rx rtune bus
10 ovrd_rtune_tx	Writes DAC_VAL[5:0] to the Tx rtune bus
dac_val	Digital value to be used for DAC

54.6.9 Resistor Tuning Control Register (clock_RTUNE_CTL)

Reset value: 16'b xxxx x000 0010 0000

This register contains resistor tuning controls.

Address: 0h base + 9h offset = 9h

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved					adc_trig	rtune_trig	rtune_dis
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	cmp_invert	dac_chop	rsc_x4	sel_atbp	pwron_lcl	frc_pwron	mode	
Reset	0	0	1	0	0	0	0	0

clock_RTUNE_CTL field descriptions

Field	Description
15–11 -	This field is reserved. Reserved
10 adc_trig	Triggers ADC conversion
9 rtune_trig	Triggers manual resistor calibration
8 rtune_dis	Disables automatic resistor recalibrations
7 cmp_invert	Inverts output of comparator (to reverse successive approximation register (SAR) feedback loop)
6 dac_chop	Polarity of chop control for DAC
5 rsc_x4	Sets x4 in rescal circuitry
4 sel_atbp	Selects atb_s_p for A/D measurement

Table continues on the next page...

clock_RTUNE_CTL field descriptions (continued)

Field	Description
3 pwron_lcl	Value of power-on to force
2 frc_pwron	Overrides internal power-on
mode	Resistor tune SAR mode: 00 Normal restune 01 ADC 10 Rx Resistor test 11 Tx Resistor test

54.6.10 ADC Output Register (clock_ADC_OUT)

Address: 0x000A

Reset value: 16'b xxxx xxxx xxxx xxxx

This register contains the results of the ADC process. A read from this register starts a new A/D conversion.

Address: 0h base + Ah offset = Ah

Bit	15	14	13	12	11	10	9	8
Read	Reserved					fresh	value	
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	value							
Write								
Reset	0	0	0	0	0	0	0	0

clock_ADC_OUT field descriptions

Field	Description
15–11 -	This field is reserved. Reserved
10 fresh	Flag indicates that a new A/D conversion result is present.
value	A/D conversion result Based on RTUNE_CTL.MODE, this value is the result of either the last conversion (MODES 0 or 1) or the current Tx/Rx cal value (MODES 3/2).

54.6.11 Spread Spectrum Phase Register (clock_SS_PHASE)

Address: 0x000B

Reset value: 16'b xxx0 0000 0000 0000

This register contains the current MPLL phase selector value.

Address: 0h base + Bh offset = Bh

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved			zero_freq	val			
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	val						dthr	
Reset	0	0	0	0	0	0	0	0

clock_SS_PHASE field descriptions

Field	Description
15–13 -	This field is reserved. Reserved
12 zero_freq	Zero frequency register Must be set for PHASE writes to not be immediately overwritten.
11–2 val	Phase value from zero reference
dthr	Bits below the useful resolution

54.6.12 JTAG Chip ID (High Bits) Register (clock_CHIP_ID_HI)

Address: 0x000C

This register contains the internal chip ID (high 16 bits) of the JTAG interface.

Address: 0h base + Ch offset = Ch

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	chip_id_hi															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

clock_CHIP_ID_HI field descriptions

Field	Description
chip_id_hi	Internal chip ID (high 16 bits)

54.6.13 JTAG Chip ID (Low Bits) Register (clock_CHIP_ID_LOW)

Address: 0x000D

This register contains the internal chip ID (low 16 bits) of the JTAG interface.

Address: 0h base + Dh offset = Dh

clock_CHIP_ID_LOW field descriptions

Field	Description
chip_id_lo	Internal chip ID (low 16 bits)

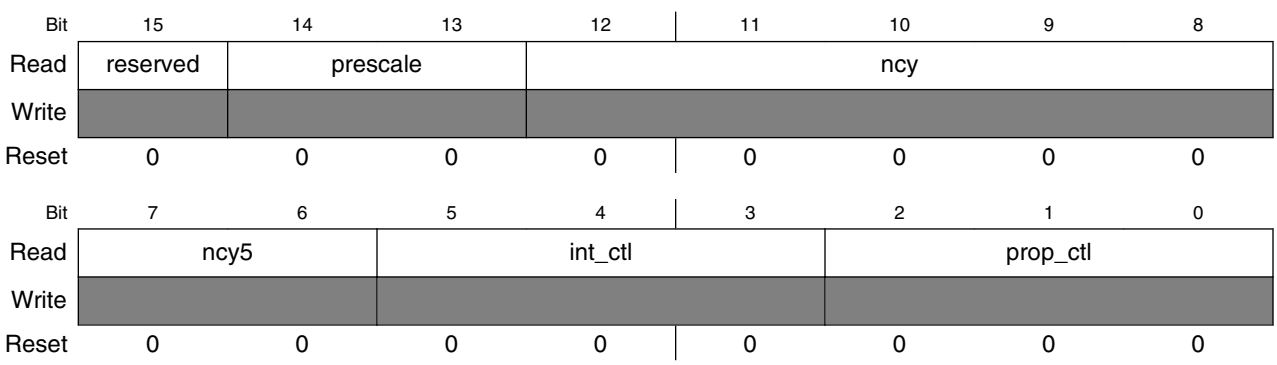
54.6.14 Frequency Status Register (clock_FREQ_STAT)

Address: 0x000E

Reset value: 16'b xxxx xxxx xxxx xxxx (depends on inputs)

This register indicates the status of frequency control inputs.

Address: 0h base + Eh offset = Eh



clock_FREQ_STAT field descriptions

Field	Description
15 reserved	This field is reserved. Always reads as 1
14–13 prescale	Prescaler control

Table continues on the next page...

clock_FREQ_STAT field descriptions (continued)

Field	Description
12–8 ncy	Divide-by-4 cycle control
7–6 ncy5	Divide-by-5 control
5–3 int_ctl	Integral charge pump control
prop_ctl	Proportional charge pump control

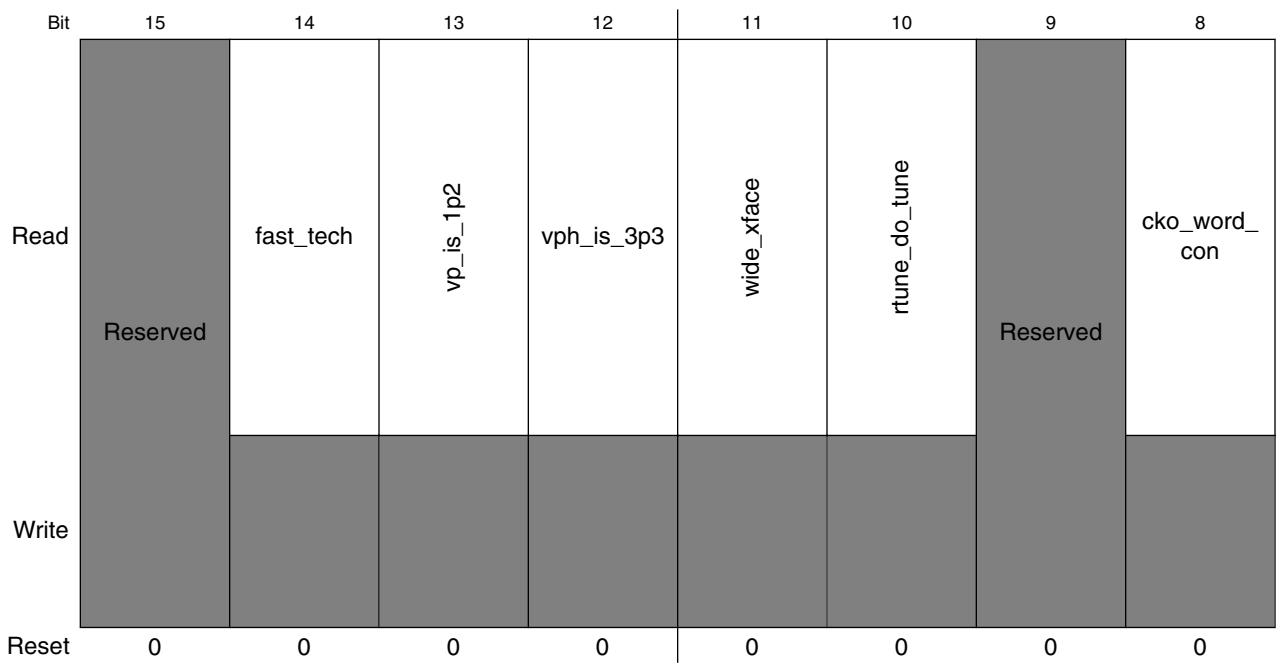
54.6.15 Control Status Register (clock_CTL_STAT)

Address: 0x000F

Reset value: 16'b xxxx xxxx xxxx xxxx (depends on inputs)

This register indicates the status of various control inputs.

Address: 0h base + Fh offset = Fh



Bit	7	6	5	4	3	2	1	0
Read	cko_word_con		cko_alive_con		mpll_ss_en	mpll_pwron	mpll_clk_off	use_refclk_alt
Write								
Reset	0	0	0	0	0	0	0	0

clock_CTL_STAT field descriptions

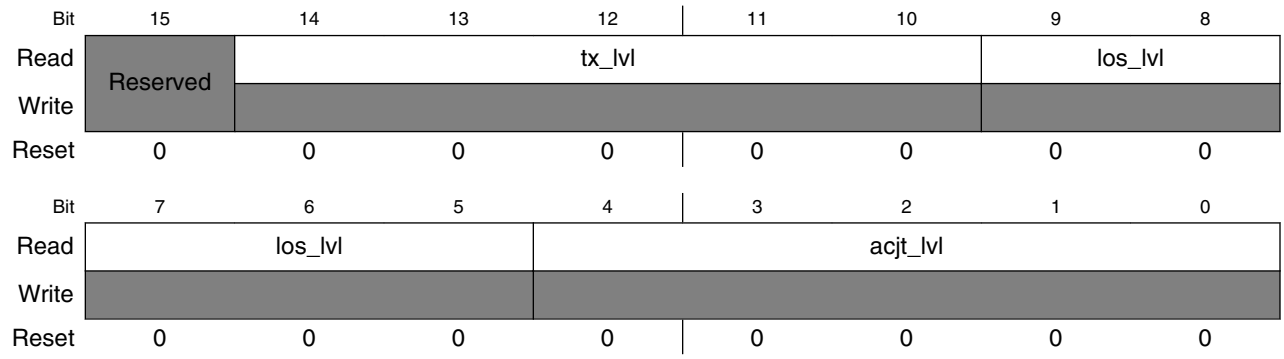
Field	Description
15 -	This field is reserved. Reserved
14 fast_tech	Technology is fast
13 vp_is_1p2	Low voltage supply is 1.2 V
12 vph_is_3p3	High voltage supply is 3.3 V
11 wide_xface	Wide interface control
10 rtune_do_tune	Manual resistor tune control
9 -	This field is reserved. Reserved
8–6 cko_word_con	cko_word MUX control
5–4 cko_alive_con	cko_alive MUX control
3 mpll_ss_en	Spread spectrum enable
2 mpll_pwron	MPLL power-on control
1 mpll_clk_off	Reference clock is off
0 use_refclk_alt	Alternate refclk is used

54.6.16 Level Status Register (clock_LVL-STAT)

Reset value: 16'b xxxx xxxx xxxx xxxx (depends on inputs)

This register indicates the status of level control inputs.

Address: 0h base + 10h offset = 10h



clock_LVL-STAT field descriptions

Field	Description
15 -	This field is reserved. Reserved
14–10 tx_lvl	Transmit level
9–5 los_lvl	Loss of Signal Detector level
acjt_lvl	ACJTAG comparator level

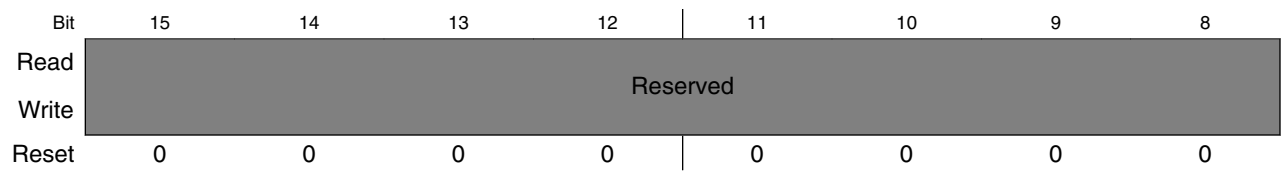
54.6.17 Creg Status Register (clock_CREG_STAT)

Address: 0x0011

Reset value: 16'b xxxx xxxx xxxx xxxx (depends on inputs)

This register indicates the status of creg control I/O.

Address: 0h base + 11h offset = 11h



Bit	7	6	5	4	3	2	1	0
Read	op_done	power_good	cr_ack	Reserved	cr_cap_addr	cr_cap_data	cr_write	cr_read
Write								
Reset	0	0	0	0	0	0	0	0

clock_CREG_STAT field descriptions

Field	Description
15–8 -	This field is reserved. Reserved
7 op_done	Operation is complete output
6 power_good	Power good output
5 cr_ack	Creg request acknowledgement
4 -	This field is reserved. Reserved
3 cr_cap_addr	Captures address request
2 cr_cap_data	Captures data request
1 cr_write	Write request
0 cr_read	Read request

54.6.18 Frequency Override Register (clock_FREW_OVRD)

Reset value: 16'b 0100 0101 0100 0111

This register contains the override of frequency control inputs.

Address: 0h base + 12h offset = 12h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	ovrd	prescale		ncy					ncy5		int_ctl			prop_ctl		
Reset	0	1	0	0	0	1	0	1	0	1	0	0	0	1	1	1

clock_FREW_OVRD field descriptions

Field	Description
15 ovrd	Enables override of all bits in this register
14–13 prescale	Prescaler control: 00 No scaling

Table continues on the next page...

clock_FREW_OVRD field descriptions (continued)

Field	Description
	01 Doubles refclk frequency 10 Halves refclk frequency 11 Reserved
12–8 ncy	Divide-by-4 cycle control MPLL Divider period = 4 x (NCY + 1) + NCY5. Valid only when NCY5 <=NCY.
7–6 ncy5	Divide-by-5 control MPLL Divider period = 4 x (NCY + 1) + NCY5. Valid only when NCY5 <=NCY
5–3 int_ctl	Integral charge pump control Integral current = (n + 1) / 8 x full_scale
prop_ctl	Proportional charge pump control Proportional current = (n + 1) / 8 x full_scale

54.6.19 Control Override Register (clock_CTL_OVRD)

Reset value: 16'b 0000 1000 0101 0100

This register contains the override of various control inputs.

Address: 0h base + 13h offset = 13h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	ovrd_static	fast_tech	vp_is_1p2	vph_is_3p3	wide_xface	rtune_do_tune	ovrd_clk	cko_word_con			cko_alive_con		mpll_ss_en	mpll_pwron	mpll_clk_off	use_refclk_alt
Reset	0	0	0	0	1	0	0	0	0	1	0	1	0	1	0	0

clock_CTL_OVRD field descriptions

Field	Description
15 ovrd_static	Overrides static controls (bits [14:10])
14 fast_tech	Technology is fast
13 vp_is_1p2	Low-voltage supply is 1.2 V
12 vph_is_3p3	High-voltage supply is 3.3 V
11 wide_xface	Wide interface control

Table continues on the next page...

clock_CTL_OVRD field descriptions (continued)

Field	Description
10 rtune_do_tune	Manual resistor tune control
9 ovrd_clk	Overrides clock controls (bits [8:0])
8–6 cko_word_con	cko_word mux control
5–4 cko_alive_con	cko_alive mux control
3 mpll_ss_en	Spread spectrum enable
2 mpll_pwron	MPLL power-on control
1 mpll_clk_off	Reference clock is off
0 use_refclk_alt	Uses alternate refclk

54.6.20 Level Override Register (clock_LVL_OVRD)

Address: 0x0014

Reset value: 16'b 0100 0010 0001 0000

This register contains the override of level control inputs.

Address: 0h base + 14h offset = 14h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	ovrd	level tx_lvl							los_lvl			acjt_lvl				
Reset	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0

clock_LVL_OVRD field descriptions

Field	Description
15 ovrd	Overrides all level controls
14–10 level tx_lvl	Transmit level
9–5 los_lvl	Loss of Signal Detector
acjt_lvl	ACJTAG comparator level

54.6.21 Creg Override Register (clock_CREG_OVRD)

Address: 0x0015 Reset value: 16'b xxxx xxx0 0100 0000 This register contains the override of creg control I/O.

Address: 0h base + 15h offset = 15h

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved							ovrd_out
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	op_done	power_good	cr_ack	ovrd_in	cr_cap_addr	cr_cap_data	cr_write	cr_read
Reset	0	1	0	0	0	0	0	0

clock_CREG_OVRD field descriptions

Field	Description
15–9 -	This field is reserved. Reserved
8 ovrd_out	Overrides outputs (bits [7:5])
7 op_done	Operation is complete output
6 power_good	Power good output
5 cr_ack	Creg request acknowledgement
4 ovrd_in	Overrides inputs (bits [3:0])
3 cr_cap_addr	Captures address request
2 cr_cap_data	Captures data request
1 cr_write	Writes request
0 cr_read	Reads request

54.6.22 MPLL Control Register (clock_MPLL_CTL)

Reset value: 16'b xxxx xx00 0000 0000

This register contains MPLL controls.

Address: 0h base + 16h offset = 16h

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved	dtb_sel1					dtb_sel0	
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	dtb_sel0			refclk_delay	dis_para_ creg	ovrd_clkdrv	clkdrv_dig	clkdrv_ana
Reset	0	0	0	0	0	0	0	0

clock_MPLL_CTL field descriptions

Field	Description
15 - Reserved	This field is reserved.
14–10 dtb_sel1	Selects wire to drive onto DTB bit 1: All other bits: Disabled 00000 Disabled 00001 mpll_gear_shift 00010 mpll_reset 00011 mpll_pwron (at analog boundary) 00100 reset_n 00101 cr_ack 00110 power_good 00111 op_done 01000 cr_read 01001 cr_write 01010 cr_cap_data 01011 cr_cap_addr 01100 rtune_do_tune 01101 cko_alive_con[0] 01110 cko_alive_con[1] 01111 cko_word_con[0] 10000 cko_word_con[1] 10001 cko_word_con[2] 10010 mpll_pwron (ASIC control) 10011 mpll_ck_off
9–5 dtb_sel0	Selects wire to drive onto DTB bit 0: All other bits: Disabled 00000 Disabled 00001 mpll_gear_shift 00010 mpll_reset 00011 mpll_pwron (at analog boundary) 00100 reset_n 00101 cr_ack 00110 power_good 00111 op_done 01000 cr_read

Table continues on the next page...

clock_MPLL_CTL field descriptions (continued)

Field	Description
	01001 cr_write 01010 cr_cap_data 01011 cr_cap_addr 01100 rtune_do_tune 01101 cko_alive_con[0] 01110 cko_alive_con[1] 01111 cko_word_con[0] 10000 cko_word_con[1] 10001 cko_word_con[2] 10010 mpll_pwron (ASIC control) 10011 mpll_ck_off
4 refclk_delay	Delays refclk output of prescaler
3 dis_para_creg	Disables parallel creg interface
2 ovrd_clkdrv	Overrides clock driver controls
1 clkdrv_dig	Value for digital clock drivers
0 clkdrv_ana	Value for analog clock drivers

54.6.23 MPLL Test Register (clock_MPLL_TEST)

Address: 0x0017

Reset value: 16'b 0000 0000 0000 0000

This register contains MPLL test controls.

Address: 0h base + 17h offset = 17h

Bit	15	14	13	12	11	10	9	8
Read Write	ovrd_ctl	gearshift_val	reset_val	meas_iv				
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	meas_iv						meas_gd	atb_sense
Reset	0	0	0	0	0	0	0	0

clock_MPLL_TEST field descriptions

Field	Description
15 ovrd_ctl	Overrides MPLL reset and gearshift controls

Table continues on the next page...

clock_MPLL_TEST field descriptions (continued)

Field	Description
14 gearshift_val	Value to override for mpll_gearshift
13 reset_val	Value to override for mpll_reset
12–2 meas_iv	Measures various MPLL controls: • Bit 2: Measures dcc_vcntrl_p on atb_sense_p • Bit 3: Measures dcc_vcntrl_m on atb_sense_m • Bit 4: Measures 1-V supply voltage on atb_sense_m • Bit 5: Measures vp_cp voltage on atb_sense_p; gd on atb_sense_m • Bit 6: Measures VCO supply voltage on atb_sense_p; gd on atb_sense_m • Bit 7: Measures clock tree supply voltage on atb_sense_p; gd on atb_sense_m • Bit 8: Measures vp16 on atb_sense_p; gd on atb_sense_m • Bit 9: Measures vref on atb_sense_p; gd on atb_sense_m • Bit 10: Measures vcntrl on atb_sense_m • Bit 11: Measures copy of bias current in oscillator on atb_force_m • Bit 12: Enables phase linearity testing of phase interpolator and VCO
1 meas_gd	Measures Ground For correct measurements, this field must be set when various meas_iv bits are set.
0 atb_sense	Hooks up ATB sense lines

54.6.24 Spread Spectrum Frequency Register (clock_SS_FREQ)

Address: 0x0018

Reset value: 16'b x011 0011 0010 1111

This register contains the frequency register override, peak frequency value, and frequency counter step values.

Address: 0h base + 18h offset = 18h

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved	freq_reg_ovrd	freq_pk					
Reset	0	0	1	1	0	0	1	1

Bit	7	6	5	4	3	2	1	0
Read Write	freq_pk	freq_cnt_init						
Reset	0	0	1	0	1	1	1	1

clock_SS_FREQ field descriptions

Field	Description
15 -	This field is reserved. Reserved

Table continues on the next page...

clock_SS_FREQ field descriptions (continued)

Field	Description
14 freq_reg_ovrd	Override control, indicating that overridden value is active
13–7 freq_pk	Peak frequency value
freq_cnt_init	Frequency counter step value. **Note:** This value is independent of the freq_pk value.

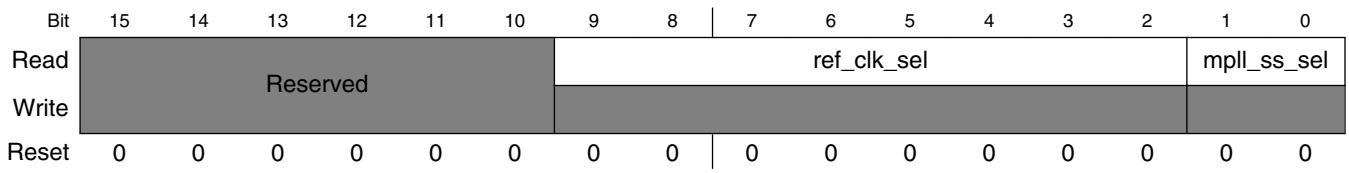
54.6.25 Clock Select Status Register (clock_SEL_STAT)

Address: 0x0019

Reset value: 16'bxxxx xxxx xxxx xxxx (depends on inputs)

This register indicates the status of the ref_clk_sel and mpll_ss_sel inputs.

Address: 0h base + 19h offset = 19h



clock_SEL_STAT field descriptions

Field	Description
15–10 -	This field is reserved. Reserved
9–2 ref_clk_sel	Reference clock select input
mpll_ss_sel	MPLL spread spectrum select input

54.6.26 Clock Select Override Register (clock_SEL_OVRD)

Address: 0x001A

Reset value: 16'b0000 0000 0000 0000

This register contains the clock select override, the ref_clk_sel override value, and the mpll_ss_sel override value.

Address: 0h base + 1Ah offset = 1Ah

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	Reserved					clk_sel_ovrd	ref_clk_sel								mpll_ss_sel	
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

clock_SEL_OVRD field descriptions

Field	Description
15–11 -	This field is reserved. Reserved
10 clk_sel_ovrd	Override control, indicating that the overridden value is active
9–2 ref_clk_sel	Reference clock select
mpll_ss_sel	MPLL spread spectrum select

54.6.27 Reset Register (clock_RESET)

Address: 0x7F3F

Reset value: 16'b xxxx xxxx xxxx xxx0

This register is a write-only register (not a real register) that resets the SATA2 PHY.

Upon writing the PHY reset bit in the reset register, the internal PHY reset is active immediately. Since the reset also affects the control register state machine, there will not be an acknowledgement of the write; that is, cr_ack will not be asserted.

NOTE

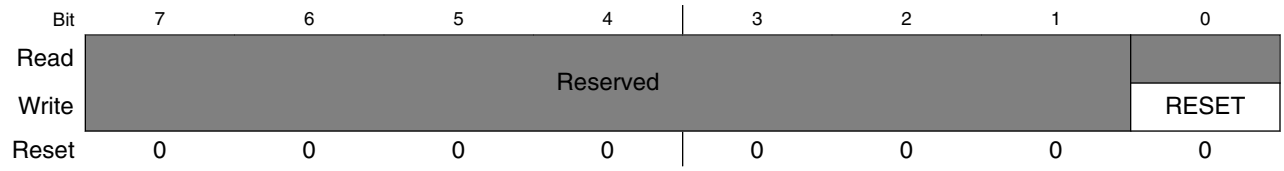
Diagnostic code should treat the *lack* of an acknowledgment of the write as a *successful* write; alternatively, it should treat the PHY *acknowledging* a write of the reset as a write *failure*. This is the opposite expectation of all other registers, where the lack is a failure and the *acknowledge* is successful.

NOTE

It is sufficient to wait 20 ref_clock cycles in order to determine that the acknowledgement has not occurred.

Address: 0h base + 7F3Fh offset = 7F3Fh

Bit	15	14	13	12	11	10	9	8
Read	Reserved							
Write								
Reset	0	0	0	0	0	0	0	0



clock_RESET field descriptions

Field	Description
15–1 -	This field is reserved. Reserved
0 RESET	Writing a 1 to this field resets the SATA2 PHY.

54.7 lane0 Memory Map/Register Definition

Register Addresses

The SATA2 PHY comprises two parts, the Clock module and one or more lane modules. Only one Clock module exists in each assembly, but there can be multiple lanes.

NOTE

SATA PHY registers are only accessible by the corresponding controller (SATA_P0_PCSR_PHYCR and SATA_P0_PCSR_PHYSR) or in debug through the JTAG port. SATA PHY is not memory mapped to processor address space, so the absolute addresses shown is the relative address and is not valid. See SATA Memory Map for more information.

Broadcast Addressing

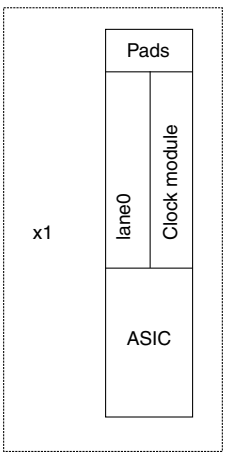
In addition to supporting writes to a single register, the register controller in the SATA2 PHY also supports broadcast writes. Broadcast writes make it easy (a single register write operation) to write the same value into N instantiations of the same register that exist in N lanes. To do a broadcast write to all lanes, simply replace the upper byte of the address with 0xA3. Note that broadcast reading of registers is not possible.

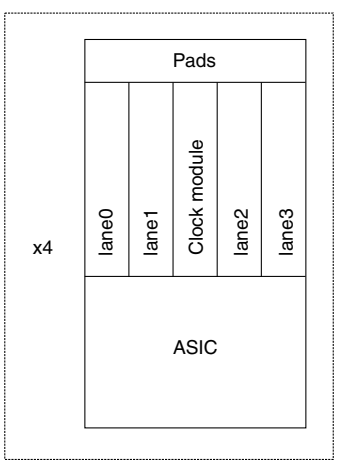
Per-Lane Register Numbering

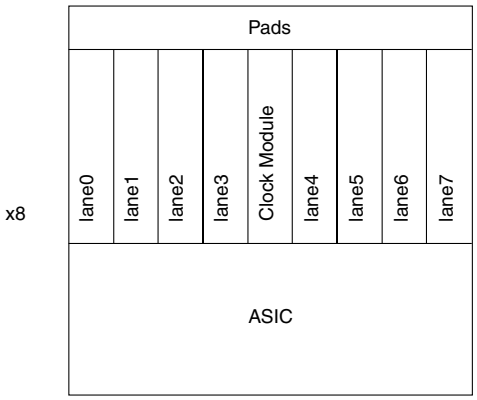
NOTE

In an N lane SATA2 PHY, there are a total of N instantiations of each register. Each register name begins with "lane0." When referring to a particular lane, the "0" is replaced with the lane

number, as in lane3.dpll.freq. The upper byte in the 16-bit address for all registers in the lane is 0x2{lane#}, where lane# is 0x0 for an x1 configuration, between 0x0 and 0x7 for an x8 configuration, and between 0x0 and 0x3 for an x4 configuration. Lane numbering always starts from 0 at the far-left side, as shown in the following three figures:







In the tables in this section, the lower byte in the 16-bit address is noted as the base address.

lane0 memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2001	Transmit Input Status Register (lane0_TX_STAT)	16	R	0000h	54.7.1/4674
2002	Receiver Input Status Register (lane0_RX_STAT)	16	R	0000h	54.7.2/4675
2003	Output Status Register (lane0_OUT_STAT)	16	R	0000h	54.7.3/4676
2004	Transmit Input Override Register (lane0_TX_OVRD)	16	R/W	0007h	54.7.4/4677
2005	Receive Input Override Register (lane0_RX_OVRD)	16	R/W	1416h	54.7.5/4678
2006	Output Override Register (lane0_OUT_OVRD)	16	R/W	0011h	54.7.6/4679
2007	Debug Control Register (lane0_DBG_CTL)	16	R/W	0000h	54.7.7/4679
2010	Pattern Generator Control Register (lane0_PG_CTL)	16	R/W	0000h	54.7.8/4682
2018	Pattern Matcher Control Register (lane0_PM_CTL)	16	R/W	0000h	54.7.9/4682
2019	Pattern Matcher Error Register (lane0_PM_ERR)	16	R/W	0000h	54.7.10/ 4683
201A	DPLL Phase Register (lane0_DPLL_PHASE)	16	R/W	0000h	54.7.11/ 4684
201B	DPLL Frequency Register (lane0_DPLL_FREQ)	16	R/W	0000h	54.7.12/ 4684
201C	Scope Control Register (lane0_SCOPE_CTL)	16	R/W	0000h	54.7.13/ 4685
201D	Receiver Control Register (lane0_RX_CTL)	16	R/W	000Fh	54.7.14/ 4685
201E	Receiver Debug Register (lane0_RX_DBG)	16	R/W	0000h	54.7.15/ 4686
2030	Receive Analog Control Register (lane0_RX_ANA_CONTROL)	16	R/W	0020h	54.7.16/ 4688
2031	Receive ATB Register (lane0_RX_ANA_ATB)	16	R/W	0000h	54.7.17/ 4688

Table continues on the next page...

lane0 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2032	Rx PLL Programming 2 Register (lane0_PLL_PRG2)	16	R/W	0000h	54.7.18/ 4689
2033	Rx PLL Programming 1 Register (lane0_PLL_PRG1)	16	R/W	02A9h	54.7.19/ 4690
2034	Rx PLL Measurement Register (lane0_PLL_PRG3)	16	R/W	0000h	54.7.20/ 4691
2035	Transmit ATB 1 Control Register (lane0_TX_ANA_ATBSEL1)	16	R/W	0000h	54.7.21/ 4692
2036	Transmit ATB 2 Control Register (lane0_TX_ANA_ATBSEL2)	16	R/W	0000h	54.7.22/ 4693
2037	Transmit Analog Control Register (lane0_TX_ANA_CONTROL)	16	R/W	0000h	54.7.23/ 4694

54.7.1 Transmit Input Status Register (lane0_TX_STAT)

Address: 0x2001

Reset value: 16'b xxxx xxxx xxxx xxxx (depends on inputs)

This register indicates the status of transmit control inputs.

Address: 0h base + 2001h offset = 2001h

Bit	15	14	13	12	11	10	9	8
Read	-	tx_edgerate			tx_atten			tx_boost
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	tx_boost		-	tx_clk_align	tx_en			tx_cko_en
Write								
Reset	0	0	0	0	0	0	0	0

lane0_TX_STAT field descriptions

Field	Description
15 -	Always reads as 1
14–13 tx_edgerate	Edge rate control
12–10 tx_atten	Attenuation amount control

Table continues on the next page...

lane0_TX_STAT field descriptions (continued)

Field	Description
9–6 tx_boost	Boost amount control
5 -	Always reads as 0
4 tx_clk_align	Command to align clocks
3–1 tx_en	Transmit enable control
0 tx_cko_en	tx_cko clock enable

54.7.2 Receiver Input Status Register (lane0_RX_STAT)

Address: 0x2002

Reset value: 16'b xxxx xxxx xxxx xxxx (depends on inputs)

This register indicates the status of receiver control inputs.

Address: 0h base + 2002h offset = 2002h

Bit	15	14	13	12	11	10	9	8
Read	-		los_ctl		dpll_reset	rx_dpll_mode		
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	rx_eq_val			rx_term_en	rx_align_en	rx_en	rx_pll_pwron	half_rate
Write								
Reset	0	0	0	0	0	0	0	0

lane0_RX_STAT field descriptions

Field	Description
15–14 -	Always reads as 1
13–12 los_ctl	LOS filtering mode control
11 dpll_reset	DPLL reset control
10–8 rx_dpll_mode	DPLL mode control
7–5 rx_eq_val	Equalization amount control

Table continues on the next page...

lane0_RX_STAT field descriptions (continued)

Field	Description
4 rx_term_en	Receiver termination enable
3 rx_align_en	Receiver alignment enable
2 rx_en	Receiver enable control
1 rx_pll_pwron	PLL power state control
0 half_rate	Digital half-rate data control

54.7.3 Output Status Register (lane0_OUT_STAT)

Address: 0x2003

Reset value: 16'b xxxx xxxx xxxx xxxx (depends on inputs)

This register indicates the status of output signals.

Address: 0h base + 2003h offset = 2003h

Bit	15	14	13	12	11	10	9	8
Read	-							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	-			tx_rxpres	tx_done	los	rx_pll_state	rx_valid
Write								
Reset	0	0	0	0	0	0	0	0

lane0_OUT_STAT field descriptions

Field	Description
15–5 -	Always reads as 1
4 tx_rxpres	Transmit receiver detection result
3 tx_done	Transmit operation is complete output
2 los	Loss of signal output
1 rx_pll_state	Current state of Rx PLL

Table continues on the next page...

lane0_OUT_STAT field descriptions (continued)

Field	Description
0 rx_valid	Receiver valid output

54.7.4 Transmit Input Override Register (lane0_TX_OVRD)

Address: 0x2004

Reset value: 16'b 0000 0000 0000 0111

This register contains the override transmitter control inputs.

Address: 0h base + 2004h offset = 2004h

Bit	15	14	13	12	11	10	9	8
Read Write	ovrd	tx_edgerate			tx_atten			tx_boost
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	tx_boost		tx_dis_align	tx_clk_align	tx_en			tx_cko_en
Reset	0	0	0	0	0	1	1	1

lane0_TX_OVRD field descriptions

Field	Description
15 ovrd	Enables override of all bits in this register
14–13 tx_edgerate	Edge rate control
12–10 tx_atten	Attenuation amount control
9–6 tx_boost	Boost amount control
5 tx_dis_align	Disables clock alignment FSM
4 tx_clk_align	Command to align clocks
3–1 tx_en	Transmit enable control
0 tx_cko_en	tx_cko clock enable

54.7.5 Receive Input Override Register (lane0_RX_OVRD)

Address: 0x2005

Reset value: 16'b x001 0100 0001 1110

This register contains the override of receiver control inputs.

Address: 0h base + 2005h offset = 2005h

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved	ovrd	los_ctl		dpll_reset	rx_dpll_mode		
Reset	0	0	0	1	0	1	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	rx_eq_val			rx_term_en	rx_align_en	rx_en	rx_pll_pwron	half_rate
Reset	0	0	0	1	0	1	1	0

lane0_RX_OVRD field descriptions

Field	Description
15 -	This field is reserved. Reserved
14 ovrd	Enables override of all bits in this register
13–12 los_ctl	LOS filtering mode control
11 dpll_reset	DPLL reset control
10–8 rx_dpll_mode	DPLL mode control
7–5 rx_eq_val	Equalization amount control
4 rx_term_en	Receiver termination enable
3 rx_align_en	Receiver alignment enable
2 rx_en	Receiver enable control
1 rx_pll_pwron	PLL power state control
0 half_rate	Digital half-rate data control

54.7.6 Output Override Register (lane0_OUT_OVRD)

Address: 0x2006

Reset value: 16'b xxxx xxxx xx01 0001

This register contains the override of output signals.

Address: 0h base + 2006h offset = 2006h

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	Reserved		ovrd	tx_rxpres	tx_done	los	rx_pll_state	rx_valid
Reset	0	0	0	1	0	0	0	1

lane0_OUT_OVRD field descriptions

Field	Description
15–6 -	This field is reserved. Reserved
5 ovrd	Enables override of all bits in this register
4 tx_rxpres	Transmit receiver detection result
3 tx_done	Transmit operation is complete output
2 los	Loss of signal output
1 rx_pll_state	Current state of Rx PLL
0 rx_valid	Receiver valid output

54.7.7 Debug Control Register (lane0_DBG_CTL)

Address: 0x2007

Reset value: 16'b x000 0000 0000 0000

This register contains debug controls.

lane0 Memory Map/Register Definition

Address: 0h base + 2007h offset = 2007h

Bit	15	14	13	12	11	10	9	8
Read	Reserved	dtb_sel1					dtb_sel0	
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	dtb_sel0			disable_rx_ck	invert_rx	invert_tx	zero_rx_ data	zero_tx_ data
Write								
Reset	0	0	0	0	0	0	0	0

lane0_DBG_CTL field descriptions

Field	Description
15 -	This field is reserved. Reserved
14–10 dtb_sel1	All other bits: Disabled Selects wire to drive onto DTB bit 1: 00000 Disabled 00001 half_rate 00010 tx_en[0] 00011 tx_en[1] 00100 tx_en[2] 00101 tx_clk_align 00110 n/a 00111 rx_pll_pwron 01000 rx_en 01001 dpll_reset 01010 rx_valid 01011 rx_pll_state 01100 los 01101 tx_done 01110 rx_ck (output to ASIC) 01111 ck_rx (PLL output) 10000 ck_los 10001 tx_ck (ASIC input) 10010 ck_tx_out (serializer output) 10011 pll_pwron (analog input) 10100 pll_reset 10101 ser_clk_kill 10110 LBERT pg strobe 10111 rx_present_p (sampled) 11000 rx_present_m (sampled)

Table continues on the next page...

lane0_DBG_CTL field descriptions (continued)

Field	Description
	11001 acjt receiver o/p from rx_p 11010 acjt receiver o/p from rx_m
9–5 dtb_sel0	All other bits: Disabled Selects wire to drive onto DTB bit 0: 00000 Disabled 00001 half_rate 00010 tx_en[0] 00011 tx_en[1] 00100 tx_en[2] 00101 tx_clk_align 00110 n/a 00111 rx_pll_pwron 01000 rx_en 01001 dpll_reset 01010 rx_valid 01011 rx_pll_state 01100 los 01101 tx_done 01110 rx_ck (output to ASIC) 01111 ck_rx (PLL output) 10000 ck_los 10001 tx_ck (ASIC input) 10010 ck_tx_out (serializer output) 10011 pll_pwron (analog input) 10100 pll_reset 10101 ser_clk_kill 10110 LBERT pattern generator strobe 10111 rx_present_p (sampled) 11000 rx_present_m (sampled) 11001 acjt receiver o/p from rx_p 11010 acjt receiver o/p from rx_m
4 disable_rx_ck	Disables rx_ck output
3 invert_rx	Inverts receive data (pre-LBERT)
2 invert_tx	Inverts transmit data (post-LBERT)
1 zero_rx_data	Overrides all receive data to zeros
0 zero_tx_data	Overrides all transmit data to zeros

54.7.8 Pattern Generator Control Register (lane0_PG_CTL)

Address: 0x2010

Reset value: 16'b xx00 0000 0000 0000 0000

This register contains pattern generator controls.

Address: 0h base + 2010h offset = 2010h

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved		pat0					
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	pat0				trigger_err	mode		
Reset	0	0	0	0	0	0	0	0

lane0_PG_CTL field descriptions

Field	Description
15–14 -	This field is reserved. Reserved
13–4 pat0	Pattern for modes 3-5
3 trigger_err	Inserts a single error into the LSB
mode	Selects a pattern to generate: 000 Disabled 001 lfsr15. X^15 + X^14 + 1 010 lfsr7. X^7 + X^6 + 1 011 Fixed word (PAT0)3'b 100 DC balanced word (PAT0, ~PAT0) 101 Fixed pattern: (000, PAT0, 3ff, ~PAT0) 110 Reserved 111 Reserved

54.7.9 Pattern Matcher Control Register (lane0_PM_CTL)

Address: 0x2018

Reset value: 16'b xxxx xxxx xxxx 0000

This register contains pattern matcher controls.

Address: 0h base + 2018h offset = 2018h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	Reserved												sync	mode		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

lane0_PM_CTL field descriptions

Field	Description
15–4 -	This field is reserved. Reserved
3 sync	To enable checking, "Synchronize pattern matcher LFSR with incoming data" must be turned on, then turned off.
mode	All other bits: Reserved Pattern to match: 000 Disabled 001 lfsr15 010 lfsr7 011 d[n] = d[n-10] 100 d[n] = !d[n-10]

54.7.10 Pattern Matcher Error Register (lane0_PM_ERR)

Address: 0x2019

Reset value: 16'b xxxx xxxx xxxx xxxx (a read resets the register)

This register is the pattern match error counter. When the clock to the error counter is turned off, reads and writes to the register are queued until the clock is turned back on.

Address: 0h base + 2019h offset = 2019h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	ov14	count														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

lane0_PM_ERR field descriptions

Field	Description
15 ov14	If this field is active, the count is multiplied by 128. If ov14 is set to 1 and count = 2^15 - 1, indicates overflow of counter.
count	Current error count If the ov14 field is active, the count is multiplied by 128.

54.7.11 DPLL Phase Register (lane0_DPLL_PHASE)

Address: 0x201A

Reset value: 16'b xxxx x000 0000 0000

This register contains the current phase selector value.

Address: 0h base + 201Ah offset = 201Ah

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	Reserved					val										dthr
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

lane0_DPLL_PHASE field descriptions

Field	Description
15–11 -	This field is reserved. Reserved
10–1 val	Phase is .UI/512 x VAL ps from zero reference
0 dthr	Bits below the useful resolution

54.7.12 DPLL Frequency Register (lane0_DPLL_FREQ)

Address: 0x201B

Reset value: 16'b xx00 0000 0000 0000

This register contains the current frequency integrator value.

Address: 0h base + 201Bh offset = 201Bh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	Reserved		val													dthr
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

lane0_DPLL_FREQ field descriptions

Field	Description
15–14 -	This field is reserved. Reserved
13–1 val	Frequency is 1.526 x VAL ppm from the reference
0 dthr	Bits below the useful resolution

54.7.13 Scope Control Register (lane0_SCOPE_CTL)

Address: 0x201C

Reset value: 16'b x000 0000 0000 0000

This register contains control bits for the per-transceiver scope portion.

Address: 0h base + 201Ch offset = 201Ch

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved	base				delay		
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	delay						mode	
Reset	0	0	0	0	0	0	0	0

lane0_SCOPE_CTL field descriptions

Field	Description
15 -	This field is reserved. Reserved
14–11 base	The bit to be sampled when mode = 2'b01
10–2 delay	Number of symbols to skip between samples
mode	Mode of counters: 00 Off 01 Sample every 10 bits (see base) 10 Sample every 11 + 10 x delay bits 11 Sample every 12 + 10 x delay bits

54.7.14 Receiver Control Register (lane0_RX_CTL)

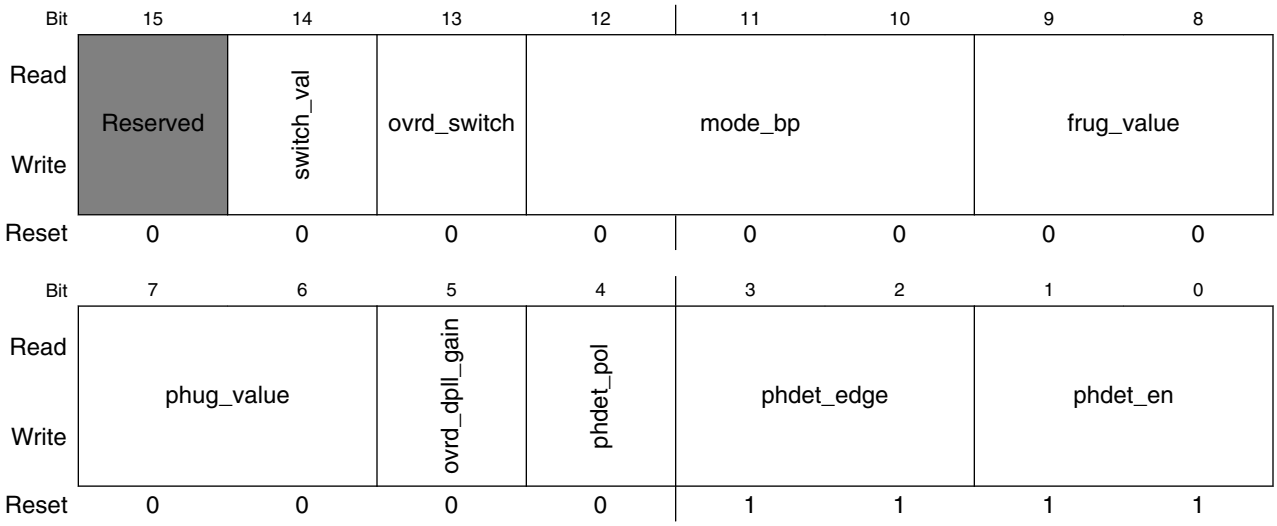
Address: 0x201D

Reset value: 16'b x000 0000 0000 1111

This register contains control bits for the receiver in the recovered domain.

lane0 Memory Map/Register Definition

Address: 0h base + 201Dh offset = 201Dh



lane0_RX_CTL field descriptions

Field	Description
15 -	This field is reserved. Reserved
14 switch_val	Value to override the data/phase MUX
13 ovrd_switch	Overrides the value of the data/phase MUX
12–10 mode_bp	Sets BP 2:0 to longer timescale (for FTS patterns): 10 Starts phase update gain (PHUG) profile at 4/3 cycles 11 Starts frequency update gain (FRUG) profile at 46/42 additional cycles 12 Ends frequency update gain (FRUG) profile at 142/110 additional cycles
9–8 frug_value	Overrides value for frequency update gain (FRUG)
7–6 phug_value	Overrides value for phase update gain (PHUG)
5 ovrd_dpll_gain	Overrides phase update gain (PHUG) and frequency update gain (FRUG) values
4 phdet_pol	Reverses polarity of phase error
3–2 phdet_edge	Edges to use for phase detection Top bit is rising edges, bottom is falling.
phdet_en	Enables phase detector Top bit is odd slicers, bottom is even.

54.7.15 Receiver Debug Register (lane0_RX_DBG)

Address: 0x201E

Reset value: 16'b xxxx xxxx 0000 0000

This register contains control bits for receiver debugging.

Address: 0h base + 201Eh offset = 201Eh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	Reserved								dtb_sel1				dtb_sel0			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

lane0_RX_DBG field descriptions

Field	Description
15–8 -	This field is reserved. Reserved
7–4 dtb_sel1	Selects wire to go on DTB bit 1 0000 Disabled 0001 ana_los 0010 los_ref_pos 0011 los_ref_neg_l 0100 sata_los 0101 los_rck 0110 eios_idle 0111 pcie_los 1000 coast_dpll 1001 misalign 1010 realign 1011 com_detect 1100 idl_detect 1101 fts_detect 1110 fts_err 1111 bp_state[1]
dtb_sel0	Selects wire to go on DTB bit 0 0000 Disabled 0001 ana_los 0010 los_ref_pos 0011 los_ref_neg_l 0100 sata_los 0101 los_rck 0110 eios_idle 0111 pcie_los 1000 coast_dpll 1001 misalign 1010 realign 1011 com_detect 1100 idl_detect 1101 fts_detect 1110 fts_err 1111 bp_state[1]

54.7.16 Receive Analog Control Register (lane0_RX_ANA_CONTROL)

Address: 0x203C

Reset value: 16'b xxxx xxxx xx10 0000

This register contains Rx control bits.

Address: 0h base + 2030h offset = 2030h

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	Reserved			rxlbi_en	rxlbe_en	rck625_en	margin_en	atb_en
Reset	0	0	1	0	0	0	0	0

lane0_RX_ANA_CONTROL field descriptions

Field	Description
15–5 -	This field is reserved. Reserved
4 rxlbi_en	Digital serial (internal) loopback enable bit When this field is set to 1, an output from the serializer is connected to the first comparator stage.
3 rxlbe_en	Wafer level (external) loopback enable bit When this field is set to 1, the lane's output (Tx) is connected to the lane's input (Rx) through pass gates.
2 rck625_en	rck625 enable bit When this field is set to 1, pll_alt_ref is driven by ck_i_p / 2.
1 margin_en	1 RWCr Margin enable bit When this field is set to 1, margining is enabled. When doing margining, ensure that you set atb_en so that atb_s_p/m are connected.
0 atb_en	ATB enable bit When this field is set to 1, internal atb_s_p,m = external atb_s_p,m.

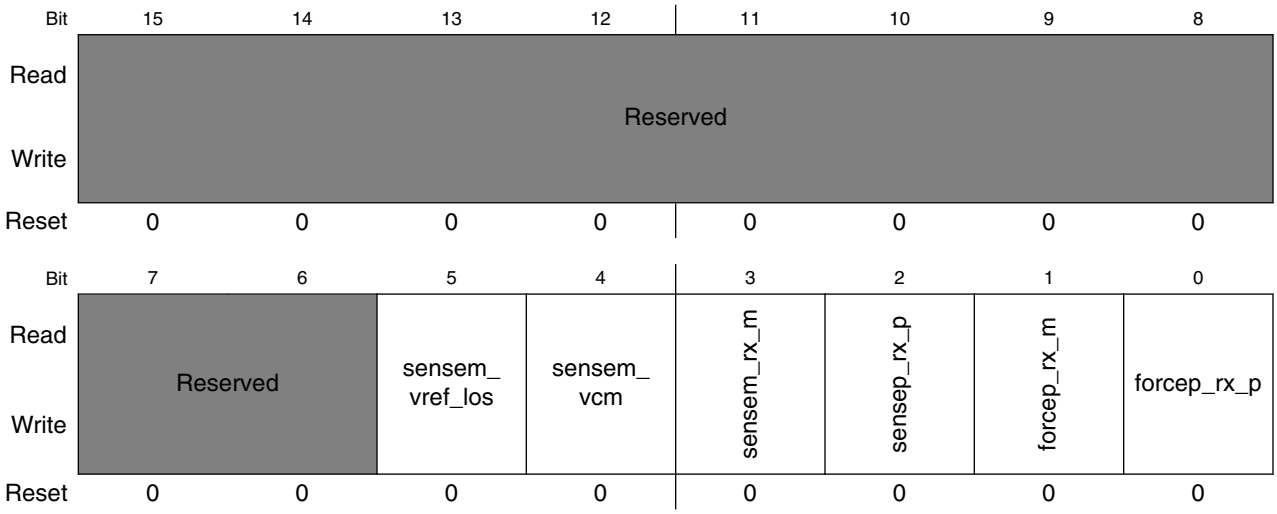
54.7.17 Receive ATB Register (lane0_RX_ANA_ATB)

Address: 0x2031

Reset value: 16'b xxxx xxxx xx00 0000

This register contains Rx ATB bits.

Address: 0h base + 2031h offset = 2031h



lane0_RX_ANA_ATB field descriptions

Field	Description
15–6 -	This field is reserved. Reserved
5 sensem_vref_los	Connects atb_s_m to vref_los (vref_rx / 14)
4 sensem_vcm	Connects atb_s_m to Rx vcm Use in margining.
3 sensem_rx_m	Connects atb_s_m to rx_m Use for measuring Rx termination.
2 sensep_rx_p	Connects atb_s_p to rx_p Use for measuring Rx termination.
1 forcep_rx_m	Connects atb_f_p to rx_m Use for measuring Rx termination.
0 forcep_rx_p	Connects atb_f_p to rx_p Use for measuring Rx termination.

54.7.18 Rx PLL Programming 2 Register (lane0_PLL_PRG2)

Address: 0x2032

Reset value: 16'b xxxx xxxx 0000 0000

This is an 8-bit programming register.

lane0 Memory Map/Register Definition

Address: 0h base + 2032h offset = 2032h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	Reserved								atb_sense_sel	frc_hcpl	hcpl_lcl	frc_pwron	pwron_lcl	frc_reset	reset_lcl	enable_test_pd
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

lane0_PLL_PRG2 field descriptions

Field	Description
15–8 -	This field is reserved. Reserved
7 atb_sense_sel	Controls proportional charge pump current: 0 Disconnects PLL from analog test bus. No PLL signals can be viewed on the ATB. 1 Enables signals internal to PLL to connect to the analog test bus.
6 frc_hcpl	Enables override of hcpl default value. Enables hcpl_lcl to control high-coupling mode.
5 hcpl_lcl	Forces coupling in VCO: Field is valid only when frc_hcpl is set to 1'b1. 0 Forces coupling in VCO to minimum. 1 Forces coupling in VCO to maximum.
4 frc_pwron	Enables override of default value of pll_pwron. Enables pwron_lcl to control PLL power-on.
3 pwron_lcl	Controls power to PLL: Field is valid only when frc_pwron is set to 1'b1. 0 Powers down PLL. 1 Supplies power to PLL.
2 frc_reset	Enables override of default value of pll_pwron. Enables pwron_lcl to control PLL power-on.
1 reset_lcl	Resets PLL: Field is valid only when frc_reset is set to 1'b1. 0 PLL is in normal mode. 1 PLL is held/placed in reset.
0 enable_test_pd	Controls phase interpolator test mode: 0 Disables phase interpolator test mode. 1 Tests phase linearity of phase interpolator and VCO.

54.7.19 Rx PLL Programming 1 Register (lane0_PLL_PRG1)

Address: 0x2033

Reset value: 16'b xxxx xx10 1010 1001

This is a 10-bit programming register.

Address: 0h base + 2033h offset = 2033h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	Reserved							sel_ rxck	prop_cntrl			int_cntrl			Reserved	
Reset	0	0	0	0	0	0	1	0	1	0	1	0	1	0	0	1

lane0_PLL_PRG1 field descriptions

Field	Description
15–9 -	This field is reserved. Reserved
8 sel_rxck	Uses recovered clock as reference to the PLL: 0 Uses MPLL output as reference to the PLL 1 Uses recovered clock as reference to PLL
7–5 prop_cntrl	Controls proportional charge pump current Proportional current = (n + 1) / 8 x full_scale Default value = 3'b101: 0.75 x full_scale
4–2 int_cntrl	Controls integral charge pump current Integral current = (n + 1) / 8 x full_scale Default value = 3'b010: 0.375 x full_scale
- 	This field is reserved. Reserved

54.7.20 Rx PLL Measurement Register (lane0_PLL_PRG3)

Address: 0x2034

Reset value: 16'b xxxx xx00 0000 0000

This is a 10-bit programming register.

Address: 0h base + 2034h offset = 2034h

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved						meas_bias	meas_vcntrl
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	meas_vref	meas_vp16	meas_ startup	meas_vco	meas_vp_cp	meas_1v	meas_ crowbar	Reserved
Reset	0	0	0	0	0	0	0	0

lane0_PLL_PRG3 field descriptions

Field	Description
15–10 -	This field is reserved. Reserved
9 meas_bias	Measures copy of bias current in oscillator on atb_force_m.
8 meas_vcntrl	Measures vcntrl on atb_sense_m. If meas_vref is also set, atb_sense_p,m measures vref - vcntrl.
7 meas_vref	Measures vref on atb_sense_p; gd on atb_sense_m. If meas_vcntrl is also set, atb_sense_p,m measures vref - vcntrl.
6 meas_vp16	Measures vp16 on atb_sense_p; gd on atb_sense_m.
5 meas_startup	Measures startup voltage on atb_sense_p; gd on atb_sense_m.
4 meas_vco	Measures VCO supply voltage on atb_sense_p; gd on atb_sense_m.
3 meas_vp_cp	Measures vp_cp voltage on atb_sense_p; gd on atb_sense_m. If meas_1v is also set, atb_sense_p,m measures vpcp - vp.
2 meas_1v	Measures 1-V supply voltage on atb_sense_m. If meas_vp_cp is also set, atb_sense_p,m measures vpcp - vp.
1 meas_crowbar	Measures crowbar bias voltage on atb_sense_p; gd on atb_sense_m.
0 -	This field is reserved. Reserved

54.7.21 Transmit ATB 1 Control Register (lane0_TX_ANA_ATBSEL1)

Address: 0x2035

Reset value: 16"b xxxx xxxx 0000 0000

This register contains Tx ATB control bits.

Address: 0h base + 2035h offset = 2035h

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	vbpf_s_p	txm_s_m	txm_f_p	txp_s_p	txp_f_p	vreg_s_m	vref_s_p	vgr_s_p
Reset	0	0	0	0	0	0	0	0

lane0_TX_ANA_ATBSEL1 field descriptions

Field	Description
15–8 -	This field is reserved. Reserved
7 vbpf_s_p	vbpf in edge rate control circuit on ATB_S_P To validate this field, set lane0.tx_ana.atbsel2.atb_en.
6 txm_s_m	txm on ATB_S_M To validate this field, set lane0.tx_ana.atbsel2.atb_en.
5 txm_f_p	txm connected to ATB_S_P For termination resistance measurements.
4 txp_s_p	txp connected to ATB_S_P To validate this field, set lane0.tx_ana.atbsel2.atb_en.
3 txp_f_p	txp connected to ATB_F_P For termination resistance measurements.
2 vreg_s_m	Regulator output voltage on ATB_S_M To validate this field, set lane0.tx_ana.atbsel2.atb_en.
1 vref_s_p	tx_vref voltage on ATB_S_P To validate this field, set lane0.tx_ana.atbsel2.atb_en.
0 vgr_s_p	Regulator gate voltage on ATB_S_P To validate this field, set lane0.tx_ana.atbsel2.atb_en.

54.7.22 Transmit ATB 2 Control Register (lane0_TX_ANA_ATBSEL2)

Address: 0x2036

Reset value: 16'b xxxx xxxx 0000 0000

This register contains Tx ATB control bits.

Address: 0h base + 2036h offset = 2036h

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	atb_en	vrefrxd_s_m	vcm_s_p	vbns_s_m	vbps_s_p	vbnf_s_m	enlpbk	en_txilpbk
Reset	0	0	0	0	0	0	0	0

lane0_TX_ANA_ATBSEL2 field descriptions

Field	Description
15–8 -	This field is reserved. Reserved
7 atb_en	RWCr 7 RWCr Connects internal and external ATB buses Required for all ATB measurements.
6 vrefrxd_s_m	Reference voltage for RX_DETECT on ATB_S_M To validate this field, set the atb_en field.
5 vcm_s_p	Vcm replica on ATB_S_P To validate this field, set the atb_en field.
4 vbns_s_m	vbps in edge rate control circuit on ATB_S_M To validate this field, set the atb_en field.
3 vbps_s_p	vbps in edge rate control circuit on ATB_S_M To validate this field, set the atb_en field.
2 vbnf_s_m	vbnf in edge rate control circuit on ATB_S_M To validate this field, set the atb_en field.
1 enlpbk	Enables Tx external loopback Ensure that internal loopback is not on.
0 en_txilpbk	Enables Tx internal loopback

54.7.23 Transmit Analog Control Register (lane0_TX_ANA_CONTROL)

Address: 0x237

Reset value: 16"b xxxx xxxx 0000 0000

This register contains Tx power state control bits.

Address: 0h base + 2037h offset = 2037h

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	frc_pwrst	en_lcl		frc_do	dataovrd_lcl	frc_beacon	bcn_lcl	Reserved
Reset	0	0	0	0	0	0	0	0

lane0_TX_ANA_CONTROL field descriptions

Field	Description
15–8 -	This field is reserved. Reserved
7 frc_pwrst	Locally forces power state When this field is set to 1, the tx_en[1:0] input is overridden by en_lcl.
6–5 en_lcl	Locally forces tx_en[1:0]: 00 Power off 01 Tx idle (slow) 10 Transmit data 11 Tx idle (fast)
4 frc_do	Forces dataovrd locally When set to 1, this field overrides the input data_ovrd value.
3 dataovrd_lcl	RWCr Local dataovrd control value To validate this field, set lane0.tx_ana.control.frc_do.
2 frc_beacon	Forces beacon to local value (bcn_lcl) When this field is set to 1, BCN_LVL overrides input value.
1 bcn_lcl	Local beacon on/off control value To validate this field, set lane0.tx_ana.control.frc_beacon.
0 -	This field is reserved. Reserved

Chapter 55
Smart Direct Memory Access Controller (SDMA)

55.1 Overview

The Smart Direct Memory Access (SDMA) controller offers highly-competitive DMA features combined with software-based virtual-DMA flexibility. It enables data transfers between peripheral I/O devices and internal/external memories.

The SDMA controller helps maximize system performance by off-loading the ARM core in dynamic data routing.

55.1.1 Block Diagram

The figure below shows a block diagram of the SDMA controller. It includes the custom RISC core along with its RAM, ROM, DMA units, and the scheduler.

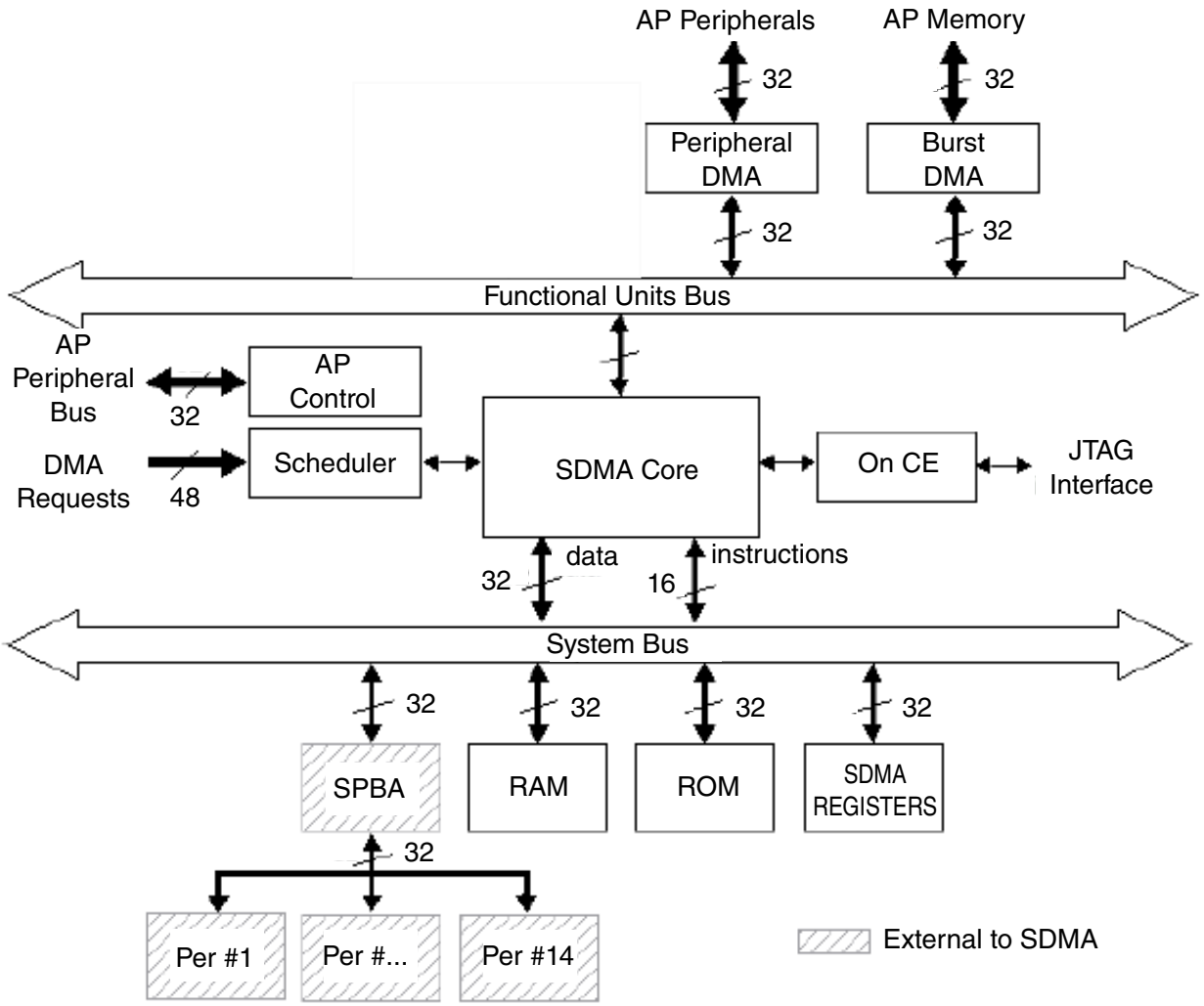


Figure 55-1. SDMA Block Diagram

The SDMA core executes short routines that perform DMA transfers; these routines are called *scripts*. The SDMA core interfaces to its own memory via the SDMA system bus. The SDMA system bus supports a 32-bit data path and a 16-bit address bus. The system bus datapath is used for both 16-bit instruction (program) memory access and 32-bit data access. DMA units interface to the core via the Functional Unit Bus and use dedicated registers to perform DMA transfers.

The SDMA memory contains a ROM and a RAM. The ROM contains startup scripts (for example, boot code) and other common utilities, which are referenced by the scripts that reside in the RAM. The internal RAM is divided into a context area and a script area (more details about this mapping are available in Instruction Memory Map and Data Memory Map).

Every transfer channel requires one context area to keep the contents of all the core and unit registers while inactive. Channel scripts are downloaded into the internal RAM by the SDMA using a dedicated channel that is started during the boot sequence. Downloads are invoked using commands and pointers provided by the ARM platform. Every channel contains a corresponding channel script located in RAM and/or ROM that can be reconfigured independently as-needed. Channel scripts can be stored in an external memory and downloaded when needed. The SDMA can be configured with any mixture of scripts to enable an endless combination of supported services.

The scheduler monitors and detects DMA requests, mapping them to channels, and mapping individual channels to a pre-configured priority. At any given point, the scheduler presents the highest priority channel that requires service to the SDMA core. A special SDMA core instruction is used to "conditionally yield" the current channel being executed to an eligible channel that requires service. If (and only if) there is an eligible channel pending, will the current channel execution be preempted.

There are two yield instructions that differently determine the eligible channels: In the first version, eligible channels are pending channels with a strictly higher priority than the current channel priority. In the second version (yieldge), eligible channels are pending channels with a priority that is greater or equal to the current channel priority. The scheduler detects devices that need service through its 48 DMA request inputs. After a request is detected, the scheduler determines the channel(s) that is (are) triggered by this request and marks it (them) as pending in the "Channel Pending (EP)" register. The priorities of all the pending channels are continuously evaluated in order to update the highest pending priority. The channel pending flag is cleared by the channel script when the transfer has completed.

The ARM platform control block contains the control registers used to configure the 32 individual channels. There are 48 Channel Enable registers, and every register maps one DMA request to any desired combination of channels. The 32 Priority registers are used to assign a programmable 1-of-7 level priority to every possible channel. This block also contains all other control registers that the ARM platform can access.

The 48 DMA requests that are connected to the scheduler come from a variety of sources. The "receive register full" and "transmit register empty" signals found in the UART and USB ports are typical examples of DMA requests that can be connected to the SDMA. These requests can be used to trigger a specific SDMA channel, or several channels.

There is an OnCE compatible debug port for product development. The OnCE includes support for setting breakpoints, single-step and trace, and register dump capability. In addition, all memory locations are accessible from the debug port.

55.1.2 Features

The following are the SDMA features:

- Multi-channel DMA supporting up to 32 time-division multiplexed DMA channels
- Hardware or software driven triggers for each channel
- 48 hardware driven triggers that can be mapped to any channel.
- Memory accesses including linear addressing, FIFO addressing and 2D addressing
- Fast context-switching with two-level, priority-based preemptive multi-tasking
- 16-bit instruction-set micro-RISC engine (the SDMA core)
- Two DMA units with some or all the following features:
  - Auto-flush and prefetch capability
  - Flexible address management (increment, decrement, and no address changes on source and destination address)
  - Misaligned data-transfer support
  - Uni-directional and bi-directional flows (copy mode)
  - Up to eight-word buffers for configurable burst transfers
- Support of byte-swapping
- An available API and library of scripts
- Little-Endian and Big-Endian modes
- Hardware handshakes for low-power entry sequence
- Security support to lock contents of the SDMA script RAM.
- 4-Kbyte ROM containing startup scripts (for example, boot code) and other common utilities that can be referenced by RAM-located scripts
- 8-Kbyte RAM area is divided into a processor context area and a code space area used to store channel scripts that are downloaded from the system memory
- Debug support, including a OnCE port, real-time monitors, and embedded cross-trigger events
- Supported clock frequencies in process:
  - Configurable clock options for the SDMA core and the ARM platform DMA units
    - 1:2 ratio with maximum of SDMA core running at ARM platform Peripheral Bus speed and DMA running at max DMA frequency.
    - 1:1 ratio when both SDMA core and ARM platform DMA clocks are set to the ARM platform Peripheral Bus speed.
- Peripheral bus interface for configuration register programming by the ARM platform
- The SDMA RISC engine (arithmetic and logic operations), which is referred to as the "SDMA core."
- An internal peripheral bus connected to the Shared Peripherals Bus Interface (SPBA) that enables access to up to 14 shared peripherals. SDMA supports 32-bit accesses to word peripherals and 16-bit accesses to half-word peripherals.

- The peripheral DMA unit that is hooked-up to the ARM platform Crossbar Switch to service ARM peripherals
- The burst DMA unit is able to perform burst accesses to the external memory
- All the DMA units are 32-bit AHB masters. They are connected to different buses, thus allowing concurrent accesses.

55.2 External Signals

The table found here describes the external signals of SDMA.

Table 55-1. SDMA External Signals

Signal	Description	Pad	Mode	Direction
SDMA_EXT_EVENT0	Event 0 signal	DISP0_DAT16	ALT4	I
		GPIO_17	ALT3	
SDMA_EXT_EVENT1	Event 1 signal	DISP0_DAT17	ALT4	I
		GPIO_18	ALT3	

55.3 Clocks

The table found here describes the clock sources for SDMA

. Please see Clock Controller Module (CCM) for clock setting, configuration and gating information. For functional information regarding module clocks, see SDMA Clocks and Low Power Modes.

Table 55-2. SDMA Clocks

Clock name	Clock Root	Description
events_sync_clk (clk)	ahb_clk_root	ARM peripheral / events clock
ips_hostctrl_clk	ipg_clk_root	Host control clock
ap_ahb_clk	ahb_clk_root	ARM platform bus clock
core_clk	ipg_clk_root	Module / Core clock
tck	-	JTAG access clock

55.4 Functional Description

The figure below shows the SDMA topology, and is composed of the following components:

- SDMA Core (SDMA Core)
- SDMA Scheduler (Scheduler)
- Functional Units:
  - Burst DMA (Burst DMA Unit)
  - Peripheral DMA (Peripheral DMA Unit)
- ARM platform Control for ARM control register access.
- Internal RAM and ROM Memory (SDMA Programming Model)
- OnCE debug Port (The OnCE Controller)

The functional unit bus provides access by the SDMA core to the DMA units. The system bus provides access to SDMA internal memory and also supports up to 14 peripherals.

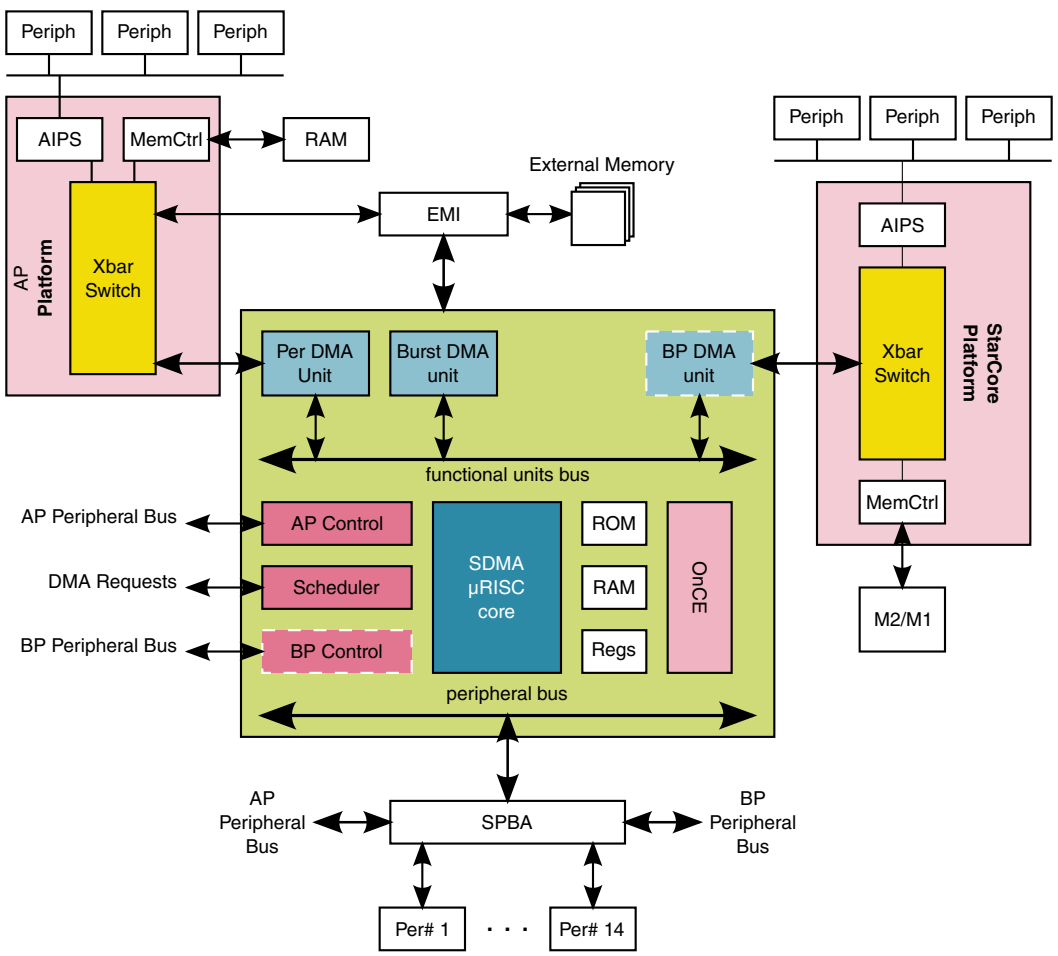


Figure 55-2. SDMA Connections

55.4.1 SDMA Core

The SDMA core is a customized RISC-like processor that is specifically developed to control DMA units and perform L1 tasks like byte-stuffing or framing.

The SDMA core incorporates on-chip debug capability using the OnCE.

The SDMA core is based on a 32-bit register architecture with 16-bit instructions. There are eight general purpose 32-bit registers, four flags (T, LM, SF, and DF), and four PCU registers (PC, RPC, SPC, and EPC) that can address 16,384 16-bit instructions.

55.4.1.1 SDMA Core Structure

The figure found here shows the structure of the SDMA core. It also shows the different registers, calculation resources, and possible data movements.

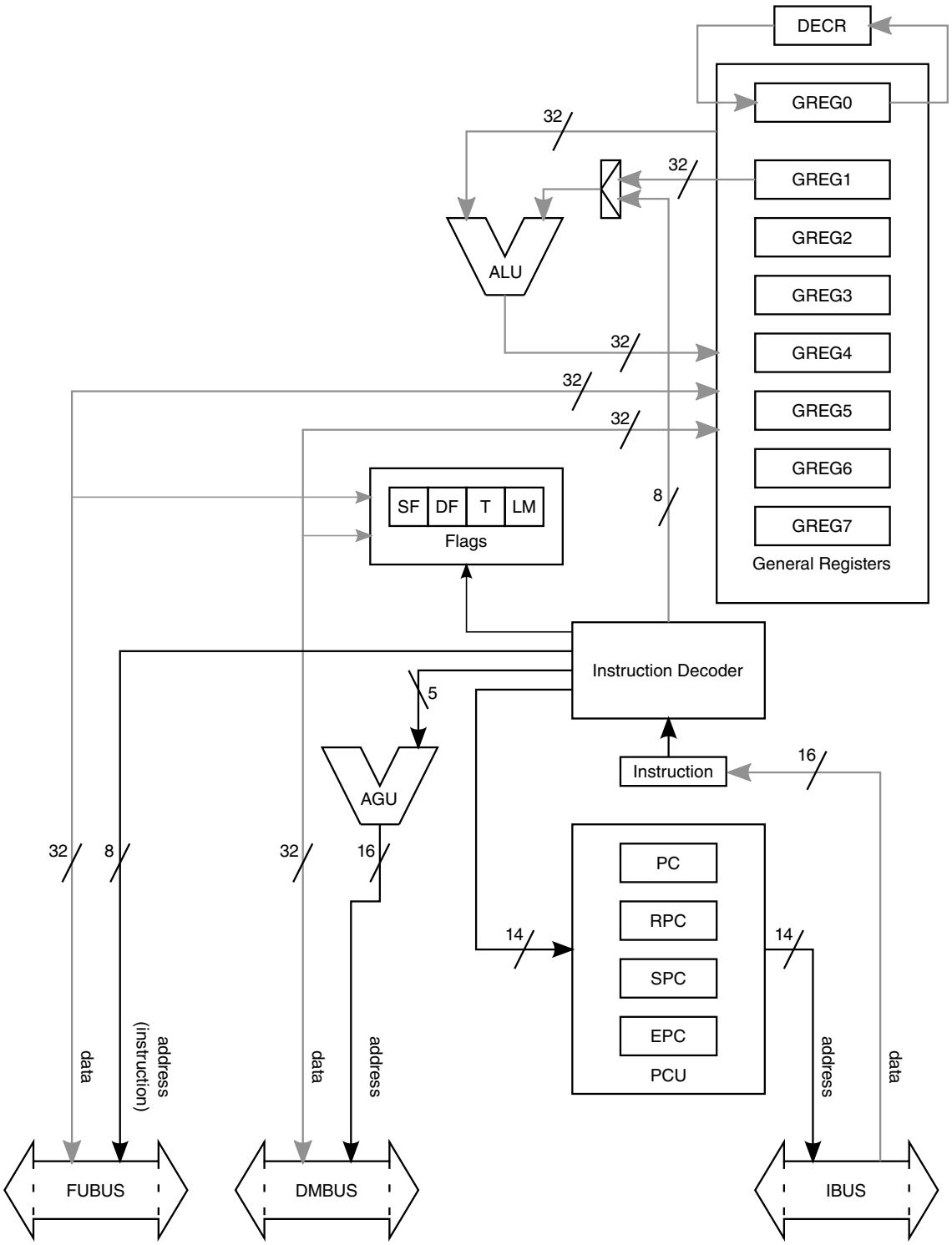


Figure 55-3. SDMA Core

- The Program Control Unit (PCU) is described in Program Control Unit (PCU). It handles the state of the core and generates the instruction fetch addresses. Instructions are retrieved from the Instruction Bus (IBUS) and stored in the SDMA

core instruction register prior to their decoding. The PCU contains the following registers:
    - The Program Counter (PC) contains the address of the current instruction.
    - The Return Program Counter (RPC) contains the address of the instruction that follows a jump to the subroutine.
    - The Start Program Counter (SPC) contains the address of the first instruction of the current hardware loop.
    - End Program Counter (EPC) contains the address of the last instruction of the current hardware loop.
- The other core registers are the general purpose registers (GREGn) and the flags.
    - The general purpose registers can be used to hold data and addresses. They can be loaded with immediate values (for example, 8-bit data that are encoded in the instruction), results of calculations that were performed with the ALU, 32-bit data that comes from the memory or peripherals via the Data Memory Bus (DMBUS), 32-bit data that comes from the DMAs via the Functional Units Bus (FUBUS) or another general purpose register. Their content can be the operands of the ALU, the data to send on either bus (DMBUS or FUBUS), or a pointer to memory (DMBUS address).
    - The general register 0 (GREG0) is also the hardware loop counter. In hardware loops, it cannot be used for any other purpose. This register uses a dedicated decrement unit (DECR) shown in Figure 55-3.
    - The flags reflect the status of operations:
        - SF and DF are set when the last load or store on either bus (FUBUS or DMBUS) received an error response.
        - LM is set when the core is executing instructions inside a hardware loop.
        - T is set when the ALU operation result was 0 or the loop counter reaches 0 (the latter is preponderant when an ALU operation is the last instruction of a hardware loop).
- The ALU has two operands: any general register and either a second general register or an immediate value. The result is always stored into the first general register. A NOP function can be utilized by moving a register's contents into itself (For example, the instruction: mov R0,R0).
- The 16-bit instructions are fetched via the instruction bus (IBUS) whose address is driven by the PC. The SDMA RAM and ROM are visible to the core as 16-bit devices through this interface.
- The memory (RAM and ROM), memory mapped registers, and external peripherals are accessed via the DMBUS. The address is always taken from a general register whose content is added to a 5-bit immediate value. This is the only available addressing mode. The DMBUS is a 32-bit data bus. Except for the peripherals that

are external to the SDMA, the address accuracy is the 32-bit word (for example, adding 1 to an address points to the next word, not the next byte).
- The functional units are accessed via the FUBUS connection. The data is exchanged with any general register, but the address (which in fact is the instruction and the selector of the functional unit) comes from an 8-bit field of the corresponding load or store.

55.4.1.2 Program Control Unit (PCU)

This part of the SDMA core is dedicated to the control of the RISC engine, as implied by the instructions that are executed. Its behavior is determined by the instruction type and the inputs of the SDMA.

It contains the PC, RPC, SPC, and EPC registers that are described in SDMA Core Structure.

55.4.1.2.1 Instruction Types

The state sequence and the delay of execution vary according to the type of the instruction. There are six possible categories of instructions, as follows:

1. Standard: Most of the instructions belong to this category, and always last 1 cycle.
2. ldf/stf: These are respectively the load and store instructions that access the functional units. They last 1+n cycles where n is the number of wait-states of the targeted functional unit.
3. ld/st: These are the load and store instructions that access the memory and peripherals. They last 1+n cycles where n is the number of wait-states of the targeted device (1 for the ROM, RAM, and memory mapped registers, 1 + the external peripheral wait-states). These instructions always last at least two cycles, but the core is able to handle them in one cycle. The first wait-state is inserted outside the core.
4. Branch: These are all the instructions that cause the Program Counter to point to another instruction other than the following one (for example, one that breaks the sequential flow). There are the absolute jumps, the conditional branches, the jump to the sub-routines, and the return from the sub-routine.
5. Loop,Modified Load or Store: The hardware loop instruction modifies the potential behavior of any load or store inside the loop (for example, when the LM flag is set). A jump may be implied after any such load or store if it received an error. The error causes an early exit of the loop, which means a jump to the instruction that follows the one that is pointed to by EPC. An additional cycle is required by the PCU to perform the jump (+1 to the ld/st/ldf/stf original execution delay). Although there is

usually an implicit jump after the last instruction of the loop when the PC goes back to SPC, this is performed at no cycle cost.

6. Done: The done, yield, or yieldge instructions are used to control channel switching. When no channel switching is performed, these instructions last a single cycle. When there is a change of channel or context switch, the delay is variable and depends on many factors (as detailed in Context Switching).

55.4.1.2.2 PCU States

The PCU state is visible through outputs of the SDMA (see Real-Time Debug Outputs) or the OnCE status register(see OnCE Status Register (OSTAT)).

The PCU state is a four-bit field that can take the values shown in the following table. Figure 55-4 shows the possible state transitions and the corresponding conditions.

Table 55-3. PCU States

Value	State	Description
0	Program	This is the usual instruction cycle.
1	Data	This state is inserted when there are wait-states during a load or a store on the data bus (ld/st type).
2	Change of Flow	This is the second cycle of any instruction that breaks the sequence of instructions (branch and done types). This state lasts only a single cycle; it is always followed by the Program state.
3	Error in Loop	This state is used when an error causes a hardware loop exit (loop-modified load or store type). This state only lasts a single cycle; it is always followed by the Program state.
4	Debug	The SDMA is stopped in debug mode.
5	Functional Unit	This state is inserted when there are wait-states during a load or a store on the functional units bus (ldf/stf type).
6	Sleep	No script is running: The core is idle after saving the last channel context.
7	Save	The context switch FSM is saving the current channel.
8	Program in Sleep	Same as Program except there is no associated channel, this state is used when instructions are executed after entering debug mode, whereas the core was in either Sleep mode.
9	Data in Sleep	This is the same as Data except there is no associated channel.
10	Change of Flow in Sleep	This is the same as Change of Flow except there is no associated channel. This state only lasts a single cycle, and is always followed by the Program in Sleep state.
11	Error in Loop in Sleep	This is the same as Error in Loop except there is no associated. channel. This state only lasts a single cycle, and is always followed by the Program in Sleep state.
12	Debug in Sleep	This is the same as Debug except the core was put in debug mode when no channel was active.
13	Functional Unit in Sleep	This is the same as Functional Unit except there is no associated channel.

Table continues on the next page...

Table 55-3. PCU States (continued)

Value	State	Description
14	Sleep after Reset	This shows that no script is running, and the core is idle after a reset. When a channel becomes active, no context is restored but the core starts its boot program located at address 0 (or the address available in register in Channel 0 Boot Address (SDMAARM_CHN0ADDR)).
15	Restore	The context switch FSM is restoring the next channel context.

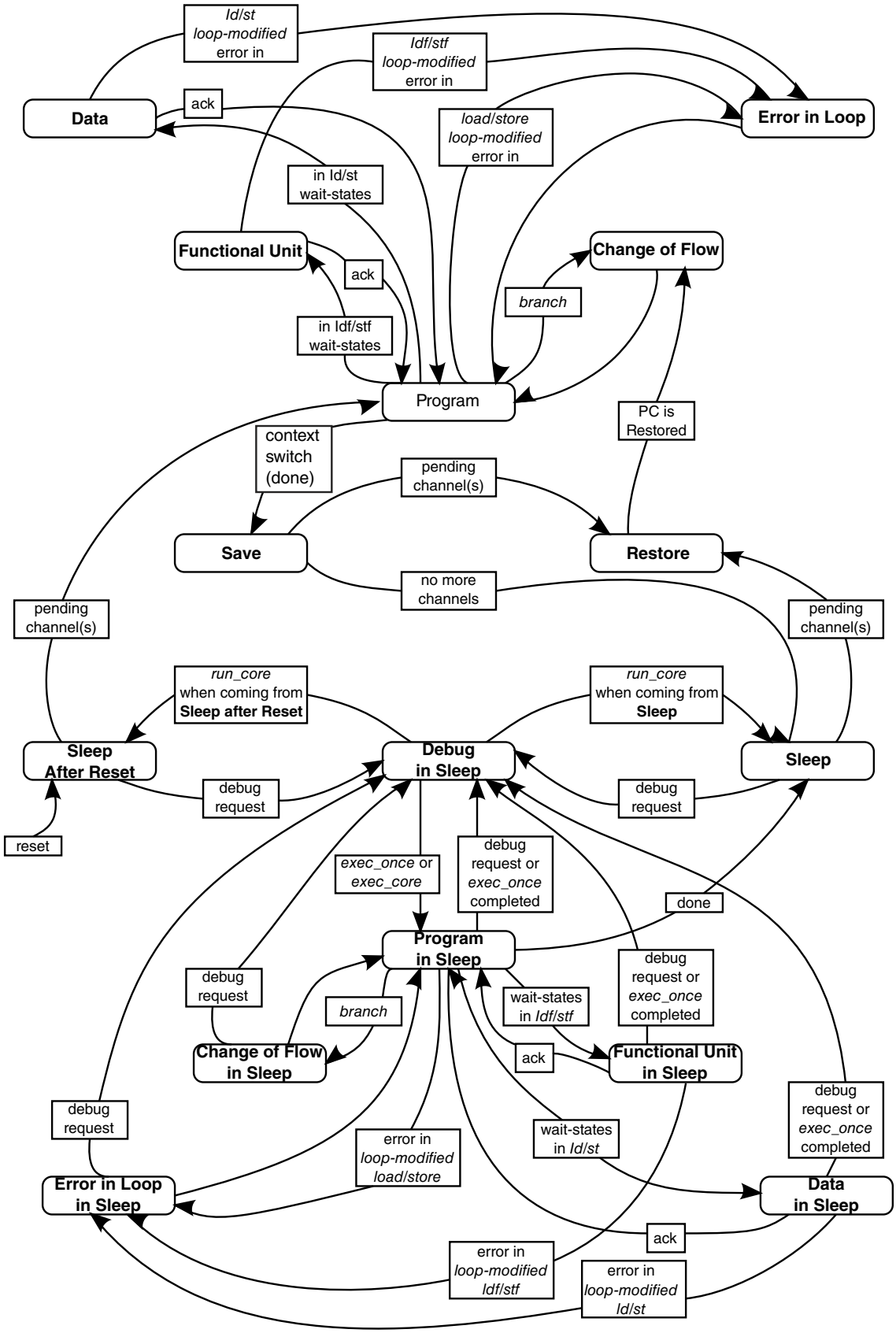


Figure 55-4. PCU State Diagram

55.4.1.3 SDMA Core Memory

The SDMA has two memory spaces: one for the instructions and one for the data. As both spaces share the same resources (ROM and RAM devices), the system bus manages possible conflicts when the core accesses the same resource for both an instruction read and a data read or write.

Program and data memory is further described in Address Space.

Instructions of 16-bit width are stored in 32-bit wide devices and can be accessed as data. The mapping is Big Endian: an even instruction address (terminated by 0) accesses the most significant part of the 32-bit data (bits [31:16]), and an odd instruction address (terminated by 1) accesses the least significant part of the 32-bit data (bits [15:0]). Instructions can be fetched out of internal ROM or RAM.

Data can be read from ROM, RAM, memory mapped registers, and external peripherals, and written to the same devices (except the ROM).

The ROM contains bootload scripts, channel scripts, and common subroutines which may be referenced by channel scripts elsewhere in the ROM or RAM.

The RAM is divided into a context area and a code space area which may be used to store channel scripts. The RAM contains undefined values after a hardware reset. Channel scripts and initial context values are downloaded into RAM using channel 0 which is reserved for bootload functions.

55.4.2 Scheduler

All channel scheduling hardware is included in the Scheduler.

55.4.2.1 Primary Functions

The scheduler is a hardware-based design used to coordinate the timely execution of 32 virtual DMA channels by the SDMA core on the basis of channel status and priority.

The scheduler performs the following functions:

- Monitors, detects, and registers the occurrence of any one of the 48 DMA requests
- Links a specific request to a channel or group of channels (channel mapping)
- Ignores requests that are not mapped to a previously configured channel
- Maintains a list of all the channels that are requesting service

- Assigns a pre-programmed priority level (1 of 7) to every channel requesting service
- Detects and flags overrun/underrun conditions

55.4.2.2 Channels and DMA Requests

55.4.2.2.1 Channels

A Virtual Channel (hereafter simply called a channel) manages a flow of data through the SDMA. Flows are typically unidirectional.

The SDMA can have up to 32 simultaneously operating channels, numbered from 0 to 31. Channel 0 is usually dedicated to control the SDMA script downloading. All the channels can be assigned by the ARM platform software.

55.4.2.2.2 DMA Requests

A DMA request is caused by externally (for example, external to the SDMA) controlled conditions (for example, UART receive FIFO reaches a threshold). The SDMA currently supports up to 48 DMA requests.

55.4.2.2.3 Mapping from DMA Requests to Channels and Priorities

A channel can stall waiting on a single DMA request. A single DMA request can awake more than one channel (in fact, any request can awake any combination of channels).

The mapping between DMA requests and channels is program-controlled. There is a storage element assigned for each of the 48 requests that contains a bitmap table of the channels that are awakened by the event.

Every channel also has a three-bit register that indicates its priority.

55.4.2.3 Scheduler Functional Description

Scheduler Overview describes the behavior of the SDMA scheduler-from the channel enabling conditions to the highest priority pending channel selection.

55.4.2.3.1 Scheduler Overview

The scheduler algorithm is built in hardware. It is provided with possibilities for the ARM platform to control its behavior.

The scheduler processes incoming DMA requests, maps detected requests to 0, one, or several channels, maintains a list of channels that are requesting service (pending channels), identifies the top priority and its associated channel, and selects the next active channel when the current channel yields.

The following figure shows a functional overview.

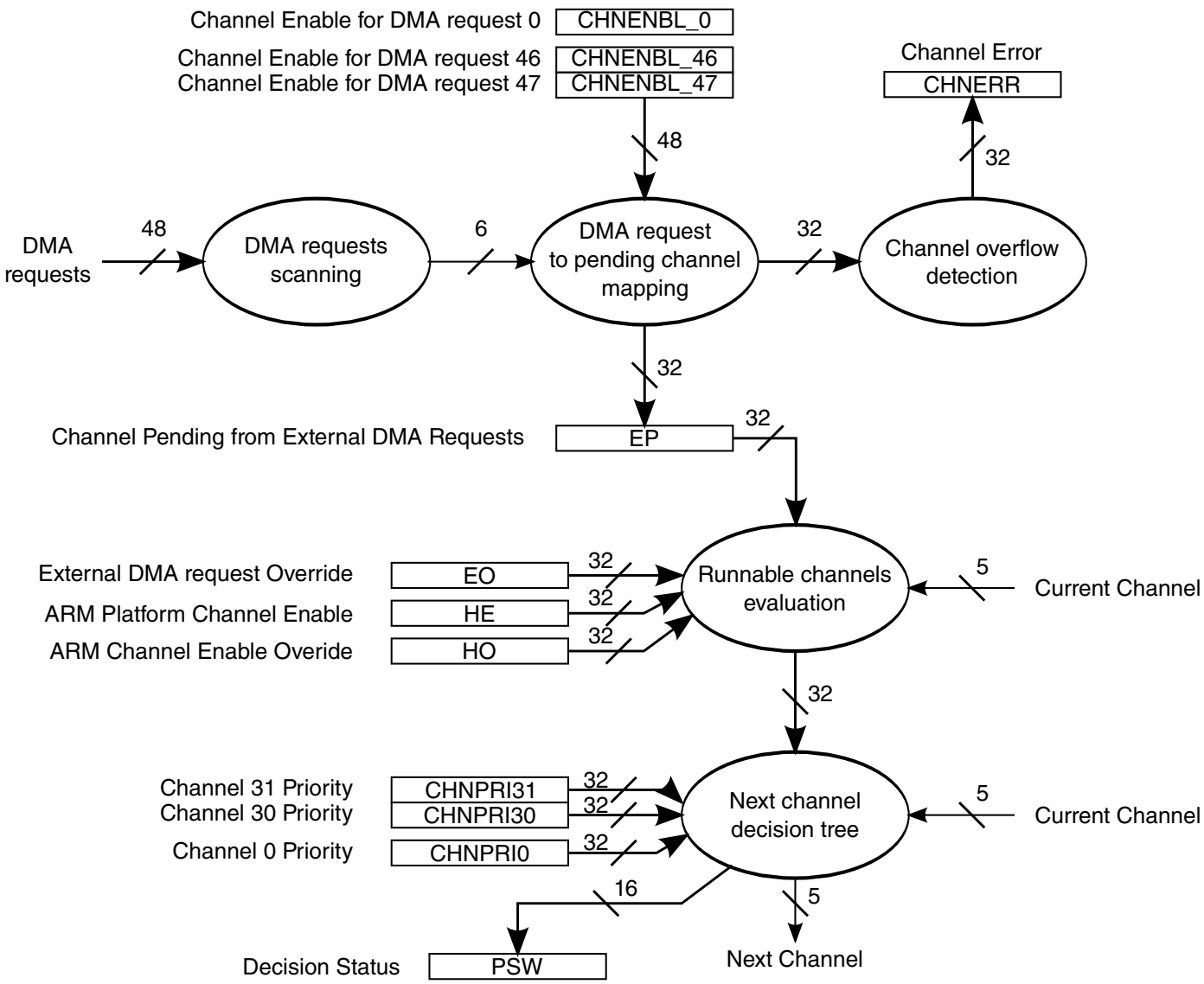


Figure 55-5. SDMA Hardware Scheduler

55.4.2.3.2 DMA Requests Scanning

The scheduler contains a 48-bit edge detection device that detects the rising edge of every DMA request and transmits the request number to the next stage.

The DMA requests are assumed to be generated on the same reference clock as the SDMA core clock; they are detected as soon as the signal goes from a 1-to-n-cycles low state to a 1-to-m-cycles high state.

This system is able to detect single-cycle pulses as well as level-based DMA requests such as a FIFO threshold crossing. In this case, the SDMA provides a memory mapped register that can be used by the channel script to monitor the DMA requests lines, and thus determines whether the data transfer is done or not done, and then continues with the transfer or closes the channel.

When several DMA requests are detected at the same time, they are forwarded to the next scheduler stage at the rate of one request per cycle. No request is lost.

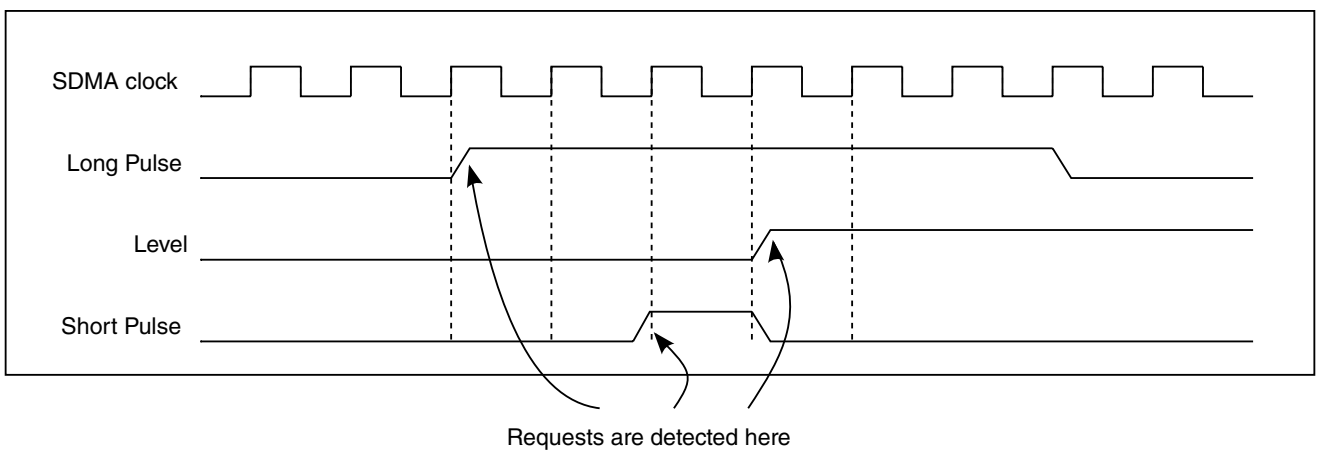


Figure 55-6. Examples of Valid DMA Requests

The DMA request inputs are connected to various sources that depend on the SoC. The exact list of DMA request inputs and their associated number is available in each respective project-specific chapter.

55.4.2.3.3 Mapping DMA Requests to Pending Channels

Whenever a DMA request is detected by the first stage, its number is used in the second stage to determine the channels that have to be activated.

This is performed with an array of 48 registers that are 32 bits wide: There are 48 Channel Enable Registers (CHNENBLn), one register per DMA request. The DMA request number selects the Channel Enable Registers, and every bit of this 32-bit register indicates that the corresponding channel must be activated when it is a 1.

This information is passed on the EP register. For every bit of the Channel Enable Register that is set, the corresponding bit of the EP register is also set, and the remaining bits of EP are left unchanged. The transformation of EP is summarized by the following equation:

EP = EP or CHNENBLn

The EP register is used to know which channels require service because they received a DMA request.

Typical contents of the CHNENBLn registers are all 0s, except for a single bit set. For example, a DMA request triggers one channel, but all 0s or several 1s are possible. One DMA request could activate several channels, and the channel execution sequence can be controlled by the channel priorities and numbers, as explained in the next sections. The following table illustrates an example configuration.

NOTE

From the table, the DMA request 0 is programmed to simultaneously trigger channels 0, 1, and 31. Also, DMA requests 30-47 are not used in this example. The remaining channels 2 to 30, are configured to be triggered by DMA requests 29 to 1, respectively.

Table 55-4. Channel Enable RAM Programming Example

DMA Request Number	Channel																																
	31																																0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table continues on the next page...

Table 55-4. Channel Enable RAM Programming Example (continued)

DMA Request Number	Channel																														
	31																														0
11	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
38	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
41	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
43	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table continues on the next page...

Table 55-4. Channel Enable RAM Programming Example (continued)

DMA Request Number	Channel																													
	31																													0
44	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
46	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
47	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.4.2.3.4 Channel Overflow

A channel overflow occurs when a DMA request requires service from channel *n* by setting bit *n* of the register EP, but this bit is already set, meaning channel *n* is already pending. This can come from an overrun/underrun condition.

This detection is possible only when the DMA requests are pulses, because a level-based DMA request stays high until it is serviced, even though an underrun or overrun condition occurs, thus preventing another edge detection of the DMA request.

The channel overflow information is saved in the 32-bit CHNERR register (1 bit per channel). You can configure the SDMA to trigger an interrupt to the ARM platform when there are 1s in CHNERR. Every bit of CHNERR is masked with the corresponding bit of INTRMASK and if it gives a 1, the corresponding bit of INTR is set, triggering the interrupt.

55.4.2.3.5 Runnable Channels Evaluation

The EP register is used in conjunction with several other 32-bit registers to determine the channels that are runnable.

Registers EO, DO, HO and HE, are controlled by the ARM platform. EP is controlled by the DMA requests and their mapping to channels.

Several channels may be runnable at any given time. The i^{th} channel is runnable if (and only if) the condition below is true:

(HE[i] or HO[i]) and (DO[i]) and (EP[i] or EO[i])

After reset, the HE[i], HO[i], EP[i], and EO[i] bits are all cleared whereas the DO[i] bits are all set. The functions associated with DO are not available for this device. When DO[i] is set, the scheduler condition becomes:

(HE[i] or HO[i]) and (EP[i] or EO[i])

The registers in these equations are controlled as follows:

- ARM platform (host) channel enable flag HE[i] may be set or cleared by the ARM platform with the HSTART and STOP_STAT registers. It can also be cleared by the i^{th} channel script.

  Typical usage is for the ARM platform to set this flag to activate the channel. The flag is cleared by the SDMA core when the transfer is done.

- Externally triggered channel pending flag EP[i] is set by the scheduler when the channel was activated by a DMA request. It can be cleared by the i^{th} channel script.
- The ARM platform channel override flag HO[i] may be set or cleared by the ARM platform. When set, it enables the i^{th} channel to run without the involvement of the ARM platform.

  Typical usage is for the ARM platform to set this flag for channels that do not need ARM platform supervision such as channels that are controlled by DMA request events (EP).

- DO should always be set to 1 so that the runnable channel evaluation considers only HO, HE, EP, and EO.
- Externally triggered channel override flag EO[i] may be set or cleared by the ARM platform. When set, it prevents the i^{th} channel from stopping and stalling on incoming peripheral DMA requests. This is the case when the channel is not handling data transfers with peripherals (for example, a memory to memory transfer).

The SDMA can clear the HE[i], and EP[i] bits by means of a done or notify instruction. The done instruction causes a reschedule; thus, enabling another channel to preempt the current one, while the notify instruction does not. The done and notify instructions can clear either HE[i] or EP[i] (never more than one at a time).

Table 55-5. Runnable Channel Selection Control

Register	Set by	Cleared By
HO	Write to HOSTOVR register	Write to HOSTOVR register
HE	Write to HSTART register	Write to STOP_STAT register or by the channel script with the done or notify instructions.
DO	Write to DSPOVR register	Write to DSPOVR register
EO	Write to EVTOVER register	Write to EVTOVER register
EP	Set by external DMA request event input.	By the channel script with the done or notify instructions

55.4.2.3.6 Next Channel Decision Tree

The next channel number is computed from the runnable channels list, the current channel number, and their respective priorities.

It is re-evaluated every cycle, but is only used when the current channel yields or terminates by executing a yield, yieldge, or done instruction.

The decision tree is based on the selection of the runnable channel that has the highest priority.

The highest priority channel is selected according to the following rules:

- Runnable channels are sorted by priority.
- If one of the channels with the highest priority had been preempted by a channel with a higher priority, but did not want to yield to a channel of the same priority (for example, it executed a yield, not a yieldge), it is elected as the next channel.
- The channels that belong to the highest priority group are sorted by their number and the channel that has the highest number in this group becomes the next channel. For example, if priorities are the same, channel 31 will be selected before channel 30.

When the current channel requires a reschedule with a yield(ge) or a done instruction, the context switch decision is based on the instruction parameter, the current channel number and priority, and the next channel number and priority. The possible cases are all listed in the following table. The grayed cells correspond to unusual cases that should not occur with a typical usage of the SDMA.

Table 55-6. Channel Switching Decision with a yield, yield(ge), or done

Instruction	Current Channel	Next Channel	Priorities Comparison	New Running Channel/Comments
yield (done 0)	Runnable	Not runnable	none	Current
	Runnable	Runnable	Current > Next	Current
			Current = Next	Current
			Current < Next	Next[1]
	Not runnable	Not runnable	none	none[2] (occurs when the channel was disabled by the ARM platform)
	Not runnable	Runnable	none	Next[1] (occurs when the channel was disabled by the ARM platform)
yieldge (done 1)	Runnable	Not runnable	none	Current
	Runnable	Runnable	Current > Next	Current
			Current = Next	Next[1]
			Current < Next	Next[1]

Table continues on the next page...

Table 55-6. Channel Switching Decision with a yield, yield(ge), or done (continued)

Instruction	Current Channel	Next Channel	Priorities Comparison	New Running Channel/Comments
	Not runnable	Not runnable	none	none[2] (occurs when the channel was disabled by the ARM platform)
	Not runnable	Runnable	none	Next[1] (occurs when the channel was disabled by the ARM platform)
done (done>1)	Not runnable	Not runnable	none	none[2]
	Runnable	Not runnable	none	Current[3] (occurs when the done instruction does not disable the channel runnable condition)
	Not runnable	Runnable	none	Next[1]
	Runnable	Runnable	none	Current[3] (occurs when the done instruction does not disable the channel runnable condition)

1. Current channel script execution is stopped, its context is saved; the next channel context is restored and its script execution resumes
2. Current channel context is saved and SDMA enters IDLE mode
3. Current channel context is saved, then restored, and the current channel script resumes execution

Finally, when the SDMA is in IDLE mode and a runnable channel is elected as the next channel, its context is immediately restored and the script execution resumes.

The *combinatorial-decision* tree supports dynamic modifications of the EP, EO, HE, HO, and DO flags as well as dynamic modifications of the channel priorities. The propagation times are detailed in Scheduler Pipeline Timing Diagram.

The decision tree status is available in the PSW register, which is continuously updated. It contains the next channel priority, the next channel number, the current channel priority, and the current channel number. When a priority is read as 0, it means the channel is not runnable.

A few examples of decisions are presented below:

- Channel 31 is running with priority 5, channels 13 and 24 are pending with the same priority 5; channel 24 is eligible as the next channel since 24 > 13.
- Channel 31 is running with priority 7, channels 13 and 24 are pending with priority 5; channel 31 is the next channel because its priority is greater than the other pending channels.
- Channels 7, 23, and 29 are pending with the same priority. Channel 7 is active and runs a yieldge; it is preempted by channel 29. After a period of time, channel 29 runs a yieldge, it is then preempted by channel 23 that is the selected channel since channel 29 is the current channel. Later, channel 23 runs a yieldge and is preempted

by channel 29. Channels 23 and 29 will go on switching after every yieldge until one of them terminates. It is only at that point that channel 7 becomes eligible again.

- Channel 11 is running with priority 3, and channel 15 is pending with priority 4. When the channel 31 script executes a yield instruction, it gets preempted by channel 15; then channels 6 and 18 with priority 3 become pending. Because channel 11 was preempted after executing a yield and there is no pending channel with a strictly greater priority, it is eligible as the next channel (although its number 11 < 18).

55.4.2.3.7 Scheduler State Diagram

The Figure 55-7 summarizes the behavior of the SDMA scheduler with details about the exact mechanism of the priority decision tree. It is important to understand the scheduler is a hardwired pipeline, which means all the stages are performed simultaneously every cycle, but a change on any given stage is reflected on the next stage after the delays presented in Scheduler Pipeline Timing Diagram.

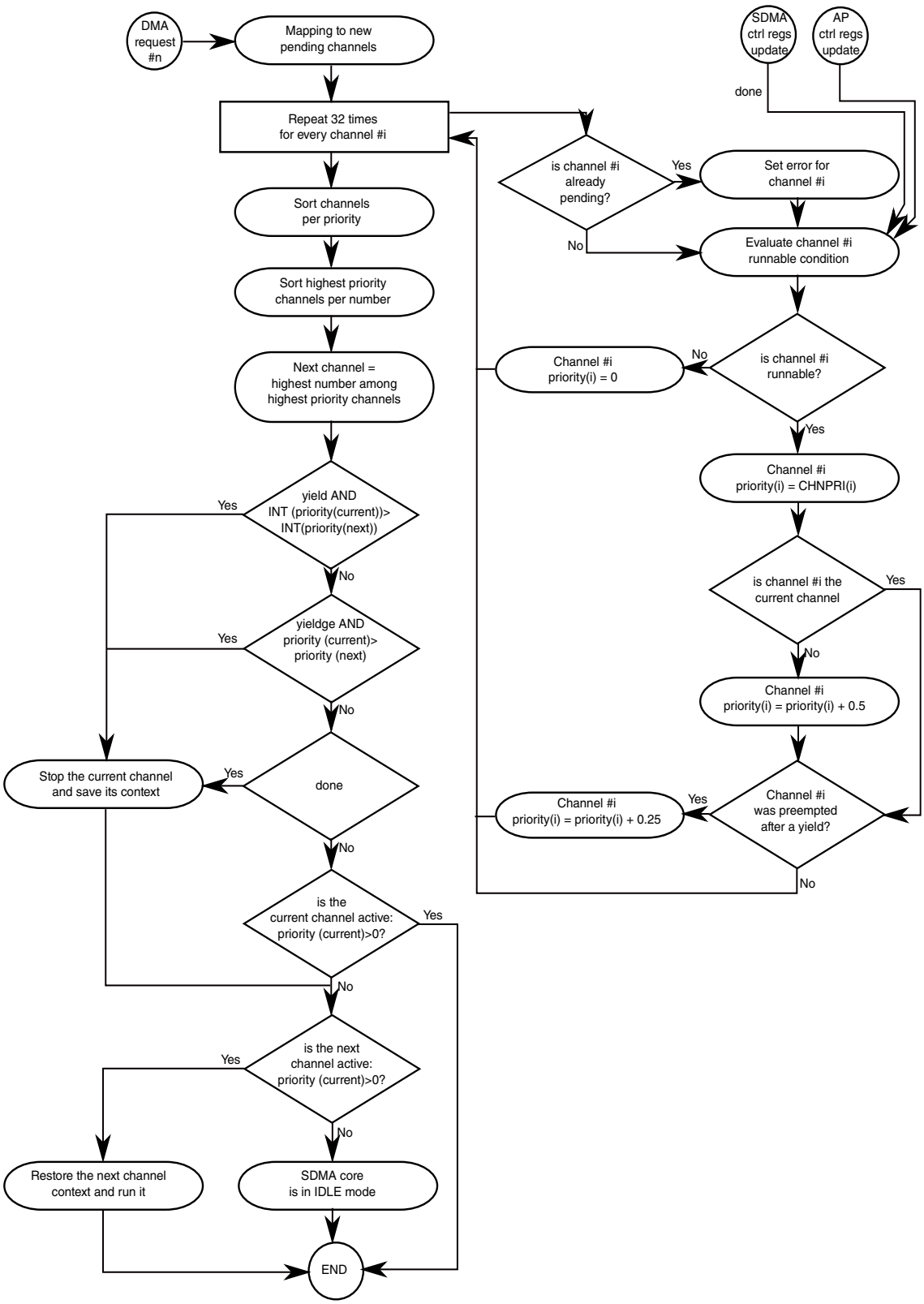


Figure 55-7. Scheduler State Diagram

55.4.2.3.8 Scheduler Pipeline Timing Diagram

The SDMA scheduler process of DMA-request and control-register modifications is not immediate.

The figure below shows the exact delays of all the tasks. The reference clock is the SDMA core clock.

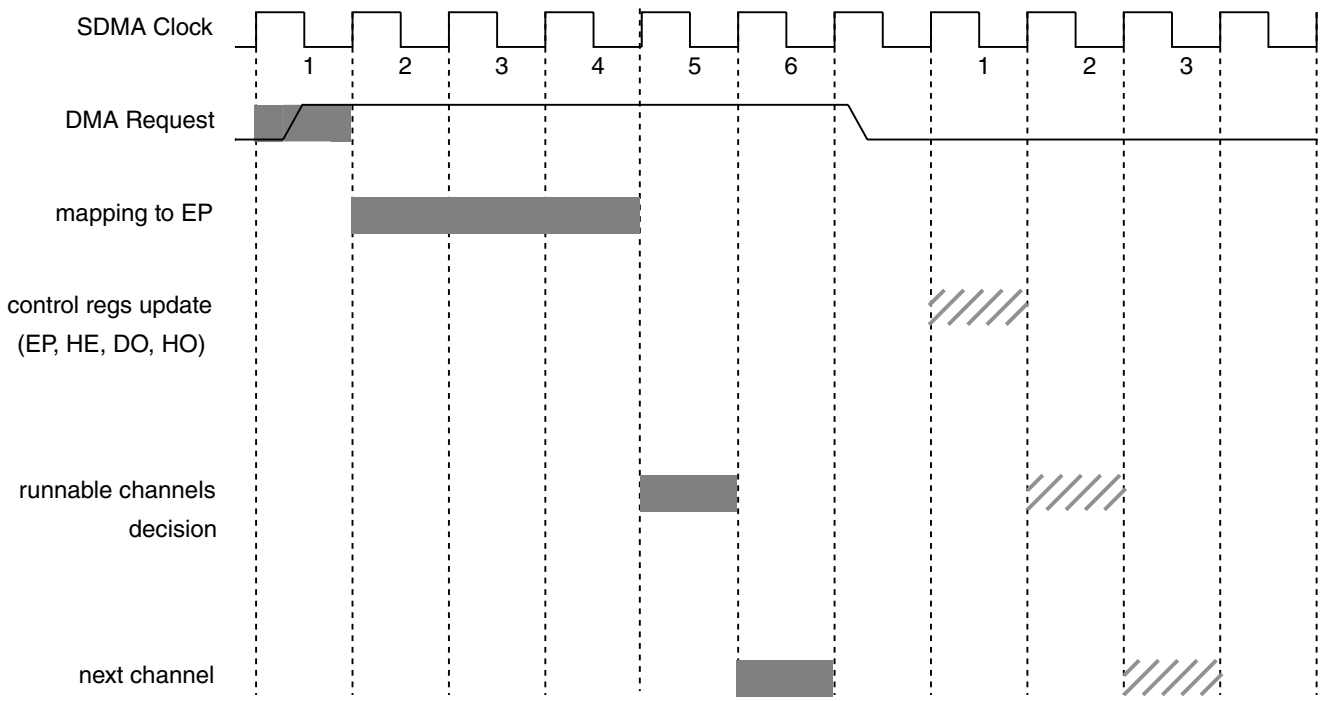


Figure 55-8. Scheduler Timing Diagram

Two numbers can be inferred from this timing diagram. First, it takes six SDMA core clock cycles to update the next channel from a DMA request. Second, it takes three SDMA core clock cycles to update the next channel from a direct modification of the condition registers (EP, DO, HE, or HO) by any processor. The processors that can modify these bits include SDMA with a done instruction or the ARM platform with a write access through the corresponding control port on their respective peripheral bus).

55.4.2.3.9 Channel-DMA Request Mapping

The 48 DMA request inputs to the SDMA scheduler are listed in project-specific chapters. Refer to the respective chapters for this information.

55.4.2.3.10 Examples: How to Start a Channel

A channel can be started when the following equation is true for channel *i*:

(HE[i] or HO[i]) and (DO[i]) and (EP[i] or EO[i])

Once this equation is true, the scheduler can start this channel according to the priority of all pending channels. Several examples of configuration are listed below:

1. To start a channel triggered by ARM platform software:
   - Initially, configure HO[i]=0, DO[i]=1, and EO[i]=1 using registers indicated in Table 55-5.
   - ARM platform software triggers the channel by writing to the HSTART register to set HE[i]=1, thereby setting the above equation true.
2. To start a channel triggered by DMA request event.
   - Initially, configure HO[i]=1, DO[i]=1, and EO[i]=0 using registers indicated in Table 55-5.
   - The DMA request is asserted to trigger the channel by setting EP[i]=1, which makes the above equation true.

55.4.2.4 Context Switching

On execution of a done or yield(ge) instruction, the current channel may be changed either because it has finished (which necessarily happens when the done instruction is executed), or it was preempted by a higher priority channel (which is possible but not systematic when the yield(ge) is executed).

Upon a channel change the SDMA goes through a context switch procedure.

When the current channel yields or ends, the context for that channel is saved into the context RAM locations for that channel. When the next channel starts running, its context is first restored from RAM.

Since context RAM is not yet initialized by reset, there will be no context restore at the beginning of the first channel (bootload channel) run after reset. It is expected that the bootload channel will be used to initialize the context for all other channels. When the bootload channel finishes running or yields, SDMA will enter its SAVE state and save that channel's context into RAM. Then, if the bootload channel is called again later, the context will be restored from RAM when the channel starts again.

The context structure for each channel is defined in Context Switching-Programming and Table 55-11. There will be one context area reserved for each channel. When a channel ends or yields, the SDMA core registers are automatically saved into the context RAM and later restored from the context RAM when the channel is next run. The total RAM space reserved for 32-channel contexts is either 3K or 4K depending on whether the SMSZ bit is set in the CHN0ADDR register, which enables an additional 8 words of scratch RAM for each context.

55.4.2.4.1 Context Switch Modes

The exact procedure to save the context of the old channel, and to restore the context of the new channel depends on the context switch mode selected by the ARM platform in the CONFIG control register.

The following are the context switch modes:

- By default, the "dynamic" context switch is set. This mode provides the most efficient context switch for an average of eight cycles to stop the current channel, save its context, restore the next channel context, and resume its execution. It consists of saving modified registers of the current channel in the background (for example, during the channel execution)-which leaves very few registers to save when the switch is decided-resuming execution of the next channel as soon as possible (for example, when the minimal set of registers is restored), and continuing the restore phase during this execution.
- In "dynamic with no loop" mode, the same principle is followed except the modified registers are only saved in the background when the loop flag is not set. This mode offers almost the same effectiveness as the previous one, but it prevents the system from accessing the RAM during loops to save power. This is the recommended mode for an efficient context-switch when the loop bodies are short.
- In "dynamic power" mode, no background saving is performed, which reduces power consumption to the minimum. The modified registers are only saved when the context switch starts. The restore phase is the same as before. This is the mode that achieves the optimal power consumption at the cost of a slower context-switch.
- In a "static" context switch, all the registers are saved when a context switch is decided, and all the registers are restored before starting the execution of the new channel. This mode enables a predictable behavior of the context switch since all the registers are restored prior to the channel start and all registers are saved after the channel termination.

NOTE

Static context mode should be used for the first channel called after reset to ensure that the all context RAM for that channel is initialized during the context SAVE phase when the channel is done or yields. Subsequent calls to the same channel or different channels may use any of the dynamic context modes. This will ensure that all context locations for the bootload channel are initialized, and prevent undefined values in context RAM from being loaded during the context restore if the channel is re-started later.

55.4.2.4.2 Context Switch Procedure

The Program Control Unit goes into the *save* state, the current context is spilled into memory, and the next channel context is restored according to the context-switch mode that was selected by the ARM platform.

The context switch procedure is as follows:

1. Load the current context's spill base address.
2. Spill the modified registers of the current channel to memory according to the selected context switch mode while the channel is running.

   On a done or yield(ge) that causes the channel preemption, the PCU goes into the *save* state. In *static* mode, all the registers are saved; whereas, in either *dynamic* mode, the registers that were modified but not yet saved are then saved, and the PCU registers and flags are finally saved.

3. Put the SDMA core into *sleep* and wait for new channels to be serviced. This step is skipped if there are pending channels when the current channel is saved.

   As soon as there is at least one pending channel, the PCU goes into its *restore* state to restore the context of the channel that was elected by the scheduler.

   Once a channel is elected, it remains the current channel until its script requests a rescheduling operation with a done or yield(ge) instruction. That means the current channel cannot be modified by the ARM platform, even if it is no more runnable or if its priority is modified.

   The ARM platform can however force a reschedule by writing the corresponding bit in the CONFIG register, which has the same effect as if the script had executed a done instruction. That feature should only be used to stop the SDMA in emergency cases.

4. Load the context base-address of the new channel.

   In "static" mode, all the registers are restored. In either "dynamic" modes, only the PCU registers are restored.

   The new channel is running. In "static" mode, no more activity regarding context restoring or saving is performed. In either "dynamic" modes, the registers are restored in the background every time an access to the context RAM is possible, and priority is given to restoring the registers that are required by the next instruction to be executed. When a register has not been restored and the next instruction needs it, this instruction gets stalled until the register was restored.

In "dynamic" and "dynamic with no loop" modes, background saving of dirty registers is performed every time an access to the context RAM is possible and allowed by the context switch mode.

NOTE

The contents of a channel context space in the context RAM depends on the selected context switch mode. In "dynamic" and "dynamic with no loop" modes, the contents of the context RAM tend to match the contents of the SDMA registers (except for the PCU registers and flags that are never saved in the background). In "dynamic power" and "static" modes, the contents of the context RAM remain unchanged until the channel terminates with a done or gets preempted.

55.4.2.4.3 Context Map in Memory

Refer to Context Switching-Programming.

55.4.3 Functional Units

The functional units are small systems that are used by the SDMA core to handle data transfers between the core and a bus domain external to the SDMA.

The SDMA core is able to control and exchange data with these systems by sending instructions and reading or writing data from/to the functional units' registers via the FUBUS. This is done with the ldf and stf instructions.

The following sections provide introductions to the available functional units. Functional Units Programming Model provides descriptions the functional units' behaviors.

55.4.3.1 Burst DMA Unit

The burst DMA unit enables the SDMA core to perform data transfers to and from the ARM platform memory.

It is optimized for accessing SDRAM-like devices. It does not provide control to assign a privilege level to the DMA access. The burst DMA unit provides the SDMA with means to do the following:

- Perform up to 8-beat read and write bursts to the ARM platform memory, which optimizes throughput when accessing SDRAM-type devices because of an internal, 36-byte FIFO
- Access the ARM platform memory at once or twice the SDMA core frequency
- Copy data from one ARM platform memory location to another ARM platform memory location at the ARM platform bus speed, which provides a very high throughput
- Control the method for addressing the ARM platform memory (automatic increment of addresses or frozen addresses-the former aimed at accessing RAM-like memory and the latter aimed at accessing single-address FIFOs)
- Enable or disable automatic prefetch when reading data from the ARM platform memory. When the prefetch mode is selected, the burst DMA automatically triggers external bursts to fill its FIFO without waiting for the SDMA core to request the corresponding data, greatly improving throughput.
- Rely on the DMA to automatically flush its FIFO content when there is enough data to generate an 8-beat burst to the ARM platform memory. Or, it forces a flush when a data transfer must terminate.
- In the former case, the SDMA core may only be stalled when it tries writing data and there is not enough room left in the FIFO. In the latter case, the core is stalled until the data is effectively written to the ARM platform memory.

  In automatic flush mode, the core receives an acknowledge that does not reflect the actual error status when the data is effectively written into the ARM platform memory. This error status is retrieved by a later access to the burst DMA. Terminating a write data transfer with a forced flush command guarantees that any bus error to the ARM platform memory is caught.

- Handle address alignment issues between the ARM platform memory map and the SDMA core data. This enables the core to read or write 32-bit data from the burst DMA, whereas the corresponding ARM platform address is not 32-bit aligned. This drastically improves the SDMA scripts' efficiency since the same loop that transfers 32 bits at a time can be used regardless of the start and end addresses in the ARM platform memory space.

This unit structure and registers are described in Burst DMA Structure and Burst DMA Registers.

55.4.3.1.1 Burst DMA Structure

The burst DMA is essentially made up of a 36-byte FIFO, address registers, and a controlling state-machine. The 36-byte FIFO enables eight-word buffering with address alignment, and the state-machine manages clock adaptation when required.

The burst DMA is depicted in the figure below.

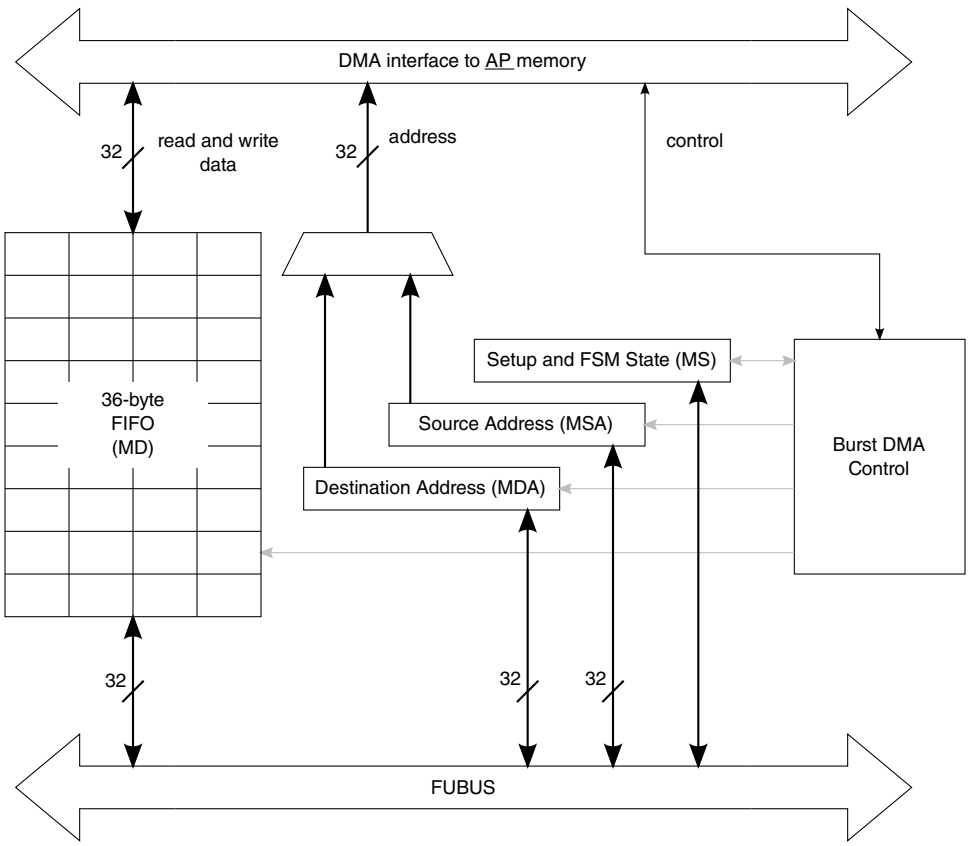


Figure 55-9. Burst DMA Structure

55.4.3.1.2 Burst DMA Registers

There are four registers, as follows, that may be accessed from the SDMA core:

- MSA (Memory Source Address) - Holds the source byte address in the ARM platform memory map for reading data from this location. This register is automatically modified every time the core reads new data from the FIFO.
- MDA (Memory Destination Address) - Holds the destination byte address in the ARM platform memory map for writing data to this location. This register is automatically modified every time the core writes new data into the FIFO.
- MD (Memory Data) - Labels the 36-byte FIFO access point: Reading a byte, halfword, or word from MD respectively retrieves the first 1, 2, or 4 bytes of the

FIFO (for example, the bytes that were stored first by the DMA state-machine when transferring data from the ARM platform memory).

- When the FIFO does not hold as many bytes as required by the SDMA core, the core is stalled until the missing bytes are read from the ARM platform memory. In the case of prefetch mode, the DMA controller decides when it should start a burst to ARM platform memory in order to reduce the risk to not have the required data for the future accesses of the core. When there is no prefetching, a burst is triggered when the required data is not available in the FIFO.

  Writing a byte, halfword, or word to MD stores 1, 2, or 4 bytes, respectively, at the end of the FIFO (for example, these bytes are transmitted to the ARM platform memory after all the other bytes that were previously stored in the FIFO). When the FIFO does not have enough room left to hold the written data, the SDMA core is stalled until a sufficient amount of FIFO contents are flushed out to the ARM platform memory. Flushing is decided by the DMA controller when there are enough bytes in the FIFO to perform the largest allowed burst to ARM platform memory (the exact size depends on the burst start address and the AHB 1 Kbyte boundary rule). However, the SDMA core has the ability to force the flushing operation at any time, for example, when at the end of the data transfer, prior to channel closure.

- MS (Memory Setup) - Contains the state of the burst DMA control, the two flags that define whether each address register is incremented after every access to the external memory, and another flag that is set when a bus error occurred.

55.4.3.1.3 Burst DMA Data Transfers

Three typical usages have been identified that involve the burst DMA: the data transfer startpoint, the endpoint, or both.

Every case requires a different procedure, as listed in the following sections:

55.4.3.1.3.1 Data Retrieval from the ARM platform Memory

The following steps retrieve data from ARM platform memory using the burst DMA unit:

- Set up the MS flags to reflect the mode for the source address (incremented or frozen according to the type of accessed device: memory or peripheral FIFO), then initialize the source address register itself (MSA).
- Read data from the FIFO using the *ldf MD* instruction as many times as needed. If an error occurred during the fetch from ARM platform memory, the DMA control tags the error status on the data and the SDMA core SF flag is set when reading this data from the FIFO.

55.4.3.1.3.2 Storing Data Into the ARM platform Memory

The following steps store data from ARM platform memory using the burst DMA unit:

- Set up the MS flags to reflect the mode for the destination address (incremented or frozen according to the type of accessed device: memory or peripheral FIFO), then initialize the destination address register itself (MDA).
- Store data into the FIFO using the *stf MD* instruction as many times as needed.
- When the transfer is finished and if the DMA worked in automatic flush mode, force the flush of the FIFO. This instruction is stalled until all the FIFO data is effectively sent to the ARM platform memory and the error status of the transfer is available in the DF flag.

55.4.3.1.3.3 Transferring Data Between Two ARM platform Memory Locations-Burst DMA Unit

The following steps copy data between two ARM platform memory locations using the burst DMA unit:

- Set up the MS flags to reflect the modes for the source and destination addresses (all the combinations are possible), then initialize the source address register (MSA) and the destination address register (MDA). Both addresses must be word-aligned.
- Use as many *stf MD* instructions with the *COPY* flag as needed. Every instruction triggers a burst read of a given number of words from the source address (this number is provided to the burst DMA via the SDMA core general purpose register, which is referenced in the *stf* instruction). Once all the data is loaded into the FIFO, the DMA empties it with a write burst of the same count to the destination address. The DMA acknowledges prior to instruction completion, which frees the SDMA core for other tasks at no delay cost.
- Once the transfer is done, there should be a final access to the burst DMA to check the error status.

55.4.3.2 Peripheral DMA Unit

The peripheral DMA unit is the second functional unit that connects the SDMA to the ARM platform memory.

Unlike the burst DMA, it does not support burst transfers and is optimized for accessing peripherals. It does not provide control to assign a privilege level to the DMA access. Its feature list comprises the following:

- Access to the ARM platform peripherals or memory at once or twice the SDMA core frequency

- Data copy from one ARM platform memory location to another ARM platform memory location at memory bus speed, improving throughput
- Control of the method for addressing the ARM platform memory (automatic increment or decrement of addresses or frozen addresses, the first ones aimed at accessing RAM-like memory and the last one aimed at accessing single-address FIFOs)
- Selectable automatic prefetch when reading data from the ARM platform memory. In prefetch mode, the peripheral DMA automatically fetches another data-without waiting for the SDMA core to request it-when its data register is empty, which improves the throughput
- Selectable automatic flush. In this mode, the SDMA core may only be stalled when it tries writing data and the previous write operation is not finished yet; whereas, in forced flush mode, the core is stalled until the data is effectively written to the ARM platform memory.
- In automatic flush mode, the core receives an acknowledge that does not reflect the actual error status when the data is effectively written into the ARM platform memory or the peripheral. This error status is retrieved by a later access to the peripheral DMA. Terminating a write data transfer with a forced flush command guarantees that any bus error to the ARM platform memory has been caught.

This unit structure and registers are described in Peripheral DMA Structure and Peripheral DMA Registers.

55.4.3.2.1 Peripheral DMA Structure

The peripheral DMA is made up of a 32-bit data register, two address registers, and a controlling state-machine. The state-machine manages clock adaptation, when required.

It is shown in the following figure.

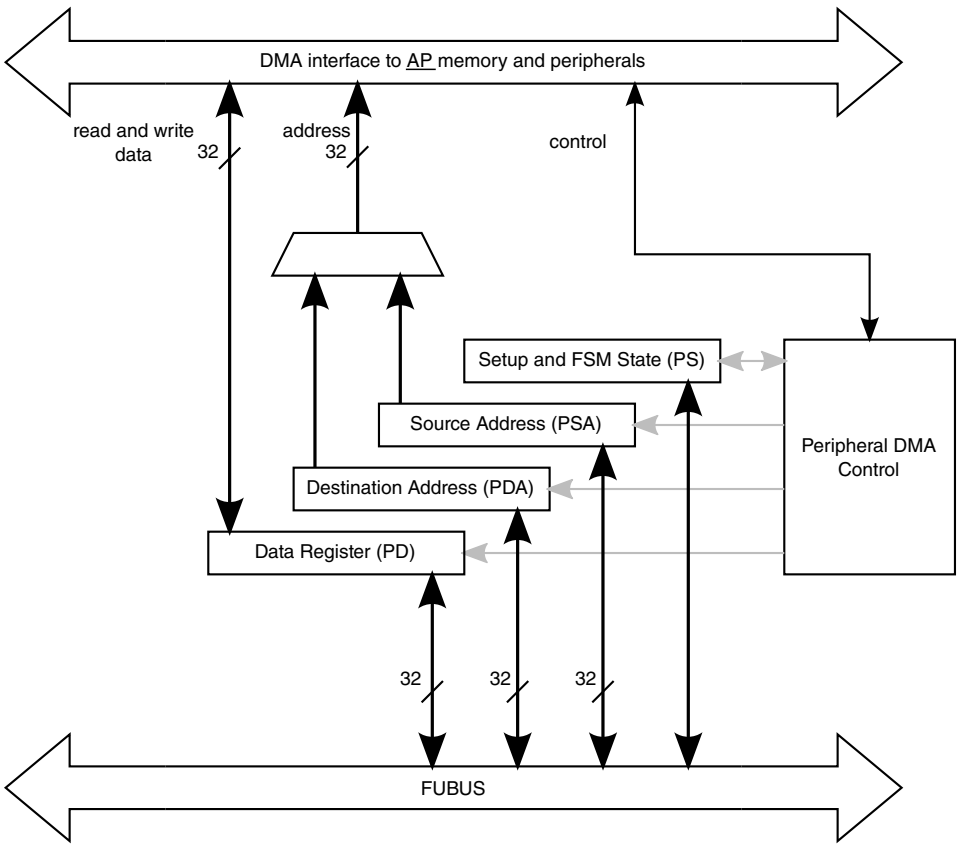


Figure 55-10. Peripheral DMA structure

55.4.3.2.2 Peripheral DMA Registers

According to Figure 55-10, the peripheral DMA has four registers that may be read or written by the SDMA core:

- *PD (Peripheral Data)* is the DMA 32-bit data register.
- *PSA (Peripheral Source Address)* holds the source byte address in the ARM platform memory map for reading data from this location. This register is automatically modified every time the core reads a new data from PD.

- *PDA (Peripheral Destination Address)* holds the destination byte address in the ARM platform memory map for writing data to this location. This register is automatically modified every time the core writes a new data into PD.
- *PS (Peripheral Setup)* contains the state of the peripheral DMA control, two configuration fields that define the way address registers are modified after every data access, two additional configuration fields that define the data size to access the source and destination devices, and another field that contains the latest transfer error status.

55.4.3.2.3 Peripheral DMA Data Transfers

There are three typical usages that involve the peripheral DMA, whether it is the data transfer start-point, endpoint, or both.

Every case requires a different procedure, as described in Data Retrieval from the ARM platform Memory or Peripheral, Storing Data into the ARM platform Memory or Peripheral, and Transferring Data Between Two ARM platform Memory Locations-Peripheral DMA Unit.

55.4.3.2.3.1 Data Retrieval from the ARM platform Memory or Peripheral

The following steps retrieve data from ARM platform memory using the peripheral DMA unit:

- Set up the PS fields to reflect the mode and data size for the source (incremented, decremented, or frozen address register; 8-bit, 16-bit, or 32-bit data transfers), then initialize the source address register itself (PSA) with an address that is aligned to the programmed data size.
- Read data from PD using the ldf PD instruction as many times as needed. If an error occurs during the fetch from the ARM platform memory or peripheral, the DMA control tags the error status on the data and the SDMA core SF flag is set when reading this data from PD.

55.4.3.2.3.2 Storing Data into the ARM platform Memory or Peripheral

The following steps store data to ARM platform memory using the peripheral DMA unit:

- Set up the PS fields to reflect the mode and data size for the destination (incremented, decremented, or frozen address register; 8-bit, 16-bit, or 32-bit data transfers), then initialize the destination address register itself (PDA) with an address that is aligned to the programmed data size.

- Store data into PD using the *stf PD* instruction as many times as needed.
- When the transfer is finished and if the peripheral DMA worked in automatic flush mode, force the flush of PD. This instruction is stalled until PD contents are effectively sent to the ARM platform memory or peripheral, and the error status of the transfer is available in the DF flag.

55.4.3.2.3.3 Transferring Data Between Two ARM platform Memory Locations-Peripheral DMA Unit

The following steps copy data between two ARM platform memory locations using the peripheral DMA unit:

- Set up the PS fields to reflect the modes and data size for the source and destination addresses (all the combinations of addressing modes are possible, but both data sizes must be identical), then initialize the source address register (PSA) and the destination address register (PDA). Both addresses must be aligned with the programmed data size.
- Use as many *stf PD* instructions with the *COPY* flag as needed. Every instruction triggers a single read from the source address; a single write of the received data immediately follows. The DMA acknowledges prior to instruction completion, which frees the SDMA core for other tasks at no delay cost.
- Once the transfer is done, there should be a final access to the peripheral DMA to check the error status.

55.4.4 SDMA Security Support

The SDMA provides support to SDMA software to block unauthorized updates to the scripts in RAM.

SDMA supports the following Security modes:

- Open Mode: has full control to load scripts and context into SDMA RAM. This is the default mode.
- Locked Mode: The ARM platform loads scripts and channel contexts at startup when it is still executing known safe software. When finished, it locks the SDMA to prevent further updates to RAM and selected registers. More details described in Locked Mode.

55.4.4.1 Locked Mode

The LOCK bit in the SDMA_LOCK register provides support for SDMA scripts to freeze RAM contents after the initial bootload routine to prevent future unauthorized updates to SDMA RAM.

After initial RAM contents are uploaded, ARM platform software can set the LOCK bit to secure the RAM contents to prevent future updates by an unauthorized. After the LOCK bit is written with a '1', the SDMA is "locked" until reset.

The LOCK bit can be read in the SDMA's internal memory map in the LOCK register (see Section SDMA LOCK (SDMAARM_SDMA_LOCK)). SDMA scripts which load information into RAM can check the value of the LOCK bit to determine if an upload to RAM is allowed. If not allowed, the script can refuse to allow the request to copy data into the RAM to continue. The exact use of the LOCK bit in SDMA scripts for security control will be described in SDMA software documentation (see SDMA Scripts).

While SDMA is locked, attempts to write to the SDMA_LOCK, CHN0ADR, ILLINSTADDR, and ONCE_ENB registers will be ignored. All registers remain readable. Writes to other registers are still allowed.

Once the SDMA is locked, the LOCK bit can only be cleared by a reset. A hardware reset will always clear the LOCK bit. A software reset initiated by writing to the RESET register will only clear the LOCK bit if the SRESET_LOCK_CLR bit in the SDMA_LOCK register is set. Since SDMA_LOCK register cannot be updated if SDMA is locked, the SRESET_LOCK_CLR bit must be configured before setting the LOCK bit. The SREST_LOCK_CLR bit will also be cleared by resets that clear the LOCK bit.

The SDMA RISC core uses the ILLINST and CHN0ADDR registers as pointers to determine where to jump to after an illegal instruction or upon boot after a reset. The LOCK bit prevents updates to these registers to protect against unauthorized changes to these pointers.

While SDMA is locked, the ONCE_ENB register cannot be written to prevent the OnCE under ARM platform control from being used to gain access to SDMA internal memory. If ARM platform control of the OnCE is enabled before setting the LOCK bit, the ARM platform can use the ONCE for debug purpose after LOCK is set.

55.4.5 OnCE and PCU Debug States

The SDMA has two different debug modes in which the OnCE performs debug instructions.

Refer to Figure 55-4 for an example of the PCU states in debug. The following are the two debug states:

- When a channel is running (that is, when CCR and CCPRI are different from 0, which can be read in the PSW register), SDMA can execute a SoftBkpt instruction from the channel script or receive a debug request. When either happens, the SDMA enters its "Classical" *Debug* state, which is described in OnCE and Real-Time Debug.
- When a channel is not running, the SDMA can be in *Sleep* state or in *Sleep after Reset* state. If a debug request is sent to the core, it enters its *Debug in Sleep* state. This debug mode works similarly to the "Classical" *Debug* state, except it returns to the original state (*Sleep* or *Sleep after Reset*) when the debug mode is left via the exec_core instruction of the OnCE. From this *Debug in Sleep* state, the SDMA can execute a program whereas no channel is running. If a new debug request is sent to the core or if a SoftBkpt is executed, it comes back to this *Debug in Sleep* state.

The OnCE is provided with several instructions that can be executed when the core is in either debug state. The following table summarizes the behavior of these OnCE debug instructions. There exists other secondary OnCE instructions that are described in OnCE and Real-Time Debug.

Table 55-7. SDMA in Debug Mode

Instruction	Debug	Debug in Sleep
exec_once	exec_once <instruction> SDMA executes the <instruction> and returns to the *Debug* state. The Program Counter (PC) is not incremented. This command must not be used with an instruction that modifies the PC value.	exec_once <instruction> SDMA executes the <instruction> and returns to the *Debug in Sleep* state. The Program Counter (PC) is not incremented. This command must not be used with an instruction that modifies the PC value.
run_core	run_core <instruction> SDMA executes the <instruction>, leaves the *Debug* state and continues executing the channel script from the position where it stopped. This command must not be used with an instruction that modifies the PC value.	run_core <instruction> SDMA executes the <instruction> and returns to its *Sleep* or *Sleep after Reset* initial state. This command must not be used with an instruction that modifies the PC value.
exec_core	exec_core <instruction> It is similar to run_core except it requires an instruction that changes the PC value (jump, branch...): the SDMA jumps to the new PC value, leaves the *Debug* state and starts executing instructions from this new PC value.	exec_core <instruction> If the previous state was *Sleep after Reset*, the SDMA returns to this state, and Chn0Addr value overrides the PC value. Otherwise, the SDMA jumps to the new PC value and starts executing instructions from this new PC.

NOTE

The feature exec_core in *Debug in Sleep* after *Sleep after Reset* was added for the Channel boot (channel 0) to allow the debugger to return to *Sleep after Reset* state with a new PC

value. The SDMA will be ready to boot at the Chn0Addr address.

55.4.6 SDMA Clocks and Low Power Modes

The SDMA receives several root clocks from the SoC clock controller block and performs adaptive clock gating to optimize its power consumption. From a user standpoint, clock gating and power mode selection are fully automatized inside the SDMA.

Root clock control is available from the SoC clock controller block.

There are numerous clock sources that are used in the SDMA. They belong to one of two possible clock domains listed in the following table, and have frequency constraints within each domain. Clocks are considered asynchronous between domains.

Within the ARM platform/SDMA clock domain, all clocks must come from the same DPLL. The ARM platform DMA interfaces (peripheral DMA and burst DMA) receive their clock from the ARM platform DMA clock source whose frequency can be once or twice the frequency of the SDMA core clock. The DMA interfaces are designed to work at the ARM platform DMA frequency, but the SDMA core is physically limited to a maximum 104 MHz frequency. Since this is lower than the maximum ARM platform DMA frequency, the SDMA core clock is tied to the ARM platform peripheral clock frequency.

The ARM platform Peripheral Bus Clock source must be an exact sub-frequency of the SDMA Core clock source (any integer value greater or equal to 1).

Table 55-8. Clocking Scheme

Clock Domain	Source Clock	Comments
ARM platform	SDMA core (SDMA main core)	Source clock for the core and all its operations; this clock is thus used by most of the SDMA sub-blocks.
	ARM platform DMA	DMA interface for the peripheral DMA and the burst DMA. It is balanced with the main clock source, and its frequency is either once or twice the main clock frequency.
	ARM platform peripheral	Connection to the ARM platform peripheral bus. It is a sub-frequency of the main clock frequency.
JTAG	TCK	Clock for JTAG access, limited to maximum of 1/8 of the SDMA core clock frequency.

The JTAG clock is sampled by the SDMA main clock to determine its rising edge. This simplifies design and clock management, but it also adds a ratio constraint between those two clocks. It is guaranteed the JTAG interface works properly when the frequency of TCK is lower than $1/8^{th}$ of the frequency of the SDMA main clock (which is about 8 MHz when the SDMA core clock frequency is 66 MHz).

55.4.6.1 Clock Gating and Low Power Modes

The SDMA automatically performs power saving without requiring user involvement. It implements two levels of automatic clock gating.

55.4.6.1.1 Coarse Clock Gating

Every sub-block clock comes from one of the five available sources, and is gated with the sub-block specific enabling condition.

The following table displays the sub-block clocks and their source. It also indicates the relationships that may exist between different sub-blocks clock enables.

Table 55-9. Sub-blocks Clocks

Sub-block	Source Clocks	Enabling Condition and Comments	Related Enabling Conditions
Core	SDMA Main Core	The core sub-block clock is running when the core is not in one of its sleep states (Sleep or Sleep after Reset) or there is a pending channel. Typically, the core sub-block clock is stopped once all the channels are processed and the core enters its sleep state. A new pending channel awakes the core sub-block clock.	None
Memories	SDMA Main Core	The clock activation only occurs during a core access.	Disabled when Core sub-block clock is disabled or no memory access in progress
Scheduler	SDMA Main Core	Its clock only runs when scheduling is needed: for example, when there are pending channels, upon reception of a DMA request, and anytime the ARM platform modifies the channel running conditions.	None
ARM platform Control	SDMA Main Core & ARM platform peripheral	The ARM platform peripheral clock is solely used to determine the frequency ratio with the SDMA main clock. The control registers' clock is based on *SDMA main clock*; it is active when the ARM platform or the SDMA modifies the contents of one of these registers.	None
Burst DMA	SDMA Main Core & ARM platform DMA	The burst DMA has two clocks: The first clock is derived from the SDMA main core clock and drives registers that are connected to the FUBUS. The second clock is derived from the ARM platform DMA clock and drives registers that are connected to the ARM platform DMA bus outside the SDMA. Both clocks are enabled	Disabled when Core sub-block clock is disabled

Table continues on the next page...

Table 55-9. Sub-blocks Clocks (continued)

Sub-block	Source Clocks	Enabling Condition and Comments	Related Enabling Conditions
		during active phases of data transfers (for example, these clocks are turned off when the burst DMA is not used by the running channel script).	
Peripheral DMA	SDMA Main Core & ARM platform DMA	The peripheral DMA has two clocks: The first clock is derived from SDMA main clock and drives registers that are connected to the FUBUS. The second clock is derived from the ARM platform DMA clock and drives registers that are connected to the ARM platform DMA bus outside the SDMA. Both clocks are enabled during active phases of data transfers (for example, these clocks are turned off when the peripheral DMA is not used by the running channel script).	Disabled when Core sub-block clock is disabled
OnCE	SDMA Main Core	The OnCE clock is derived from main source clock. It is disabled by default. In order to use the OnCE, its clock must be explicitly turned on, either by enabling the OnCE access from the ARM platform peripheral bus (register ONCE_ENB), or by driving the clk_gating_off input pin high. This is a SDMA input whose driver depends on the SoC implementation (typically a JTAG controller). The OnCE also receives the TCK input, which is the JTAG clock. It does not use it as a functional clock; the TCK input is sampled instead. Refer to Synchronization Implementation.	When enabled, all other clocks are systematically on (clock gating is off)

55.4.6.1.2 Refined Clock Gating

The SDMA implements a second level of clock gating on a register-per-register basis.

Unlike the first level that covers all the SDMA flip-flops, except the synchronizers (only five flip-flops are always running), the second level is only available for eligible registers, which amounts to about 90% of the SDMA flip-flops.

These gated registers are only clocked when the hardware logic detects a new data loading. This additional gating further reduces dynamic power consumption.

55.4.6.1.3 Low Power Modes and User Control

Power savings are automatically managed by the SDMA hardware without any user involvement; however, one can distinguish three different power modes: SLEEP, RUN, and DEBUG.

The following table describes these modes, and shows how to switch from one mode to another.

Table 55-10. Power Modes

Power Mode	Sub-blocks							Comments
	Core	Memories	Scheduler	ARM platform Control	Burst DMA	Peripheral DMA	OnCE	
SLEEP	off[1]	off	wait[2]	wait	off	off	off	Set when the PCU state is either *Sleep* or *Sleep after Reset* and the SDMA is not in DEBUG mode. This is the default mode after reset.
RUN	on[3]	wait	wait	wait	wait	wait	off	Set for the other PCU states that are reachable out of debug: *Program*, *Data*, *Change of Flow*, *Error in Loop*, *Debug*, *Functional Unit*, *Save,* or *Restore*.
DEBUG	on	on	on	on	on	on	on	Set regardless of the PCU state when clock gating is turned off to use the OnCE features (either *clk_gating_off* pin high or ONCE_ENB[0] set).

1. *off*: no clock
2. *wait*: only clocked when accessed or stimulated
3. *on*: clock is always running

It is possible to control the SDMA power mode. The procedures to force the SDMA into either mode are described in SLEEP Mode.

55.4.6.1.3.1 SLEEP Mode

This is the default mode after reset; therefore, resetting the SDMA forces this mode.

However, the common procedure is as follows:

- Ensure the *clk_gating_off* pin is low and ONCE_ENB[0] is cleared.
- Disable all channels (via the STOP_STAT control register, and the HO, DO, EO if necessary).
- Wait for the active channels to complete or force a reschedule via the reschedule bit in the RESET register.
- The SDMA is in SLEEP mode making it possible to completely shut off its clock from the chip level clock controller using the procedure described in Stop Mode Response.

55.4.6.1.3.2 RUN Mode

This is the default mode when a channel is running:

- Ensure the *clk_gating_off* pin is low and ONCE_ENB[0] is cleared.
- Activate at least one channel (via the HSTART control registers, a DMA request, and/or the HO, DO, EO register bits).

55.4.6.1.3.3 DEBUG Mode

The DEBUG mode must be set when one needs to use the debugging facilities of the SDMA.

- Ensure the SDMA clocks are running from the CCM.
- Set the *clk_gating_off* pin high or use the SDMA to set ONCE_ENB[0].

55.4.6.1.4 Stop Mode Response

The SDMA receives a stop request from the chip level clock controller. This request may be asserted when the chip enters the stop low power mode.

If the SDMA is running when the request is received, then the SDMA will complete all pending channels before returning to the SLEEP state. The SDMA sends an acknowledgement to the clock controller when the SLEEP state is entered indicating that the SDMAs clocks can be turned off.

55.4.6.2 Reset

After reset (either received from the reset block or a software reset required by the ARM platform), the SDMA is in IDLE mode. It will start its boot code located at address 0 once a channel is activated.

Activating a channel can be done by the ARM platform after programming a positive priority and setting the channel bit in the EVTPEND register.

There will not be a context RESTORE for the first channel (bootload channel) called after a reset because the context data in RAM has not been initialized. Static context mode should be used for the first channel called after reset to ensure that the all context RAM for that channel is initialized. Subsequent calls to the same channel or different channels may use any of the dynamic context modes

55.4.7 Software Interface

Appendix A fully describes the SDMA Application Programming Interface (API).

55.4.8 Initialization Information

This section discusses the following:

- Hardware Reset
- Channel Script Execution
- Initialization and Script Execution Setup Sequence

55.4.8.1 Hardware Reset

After reset, the program RAM, context RAM, data RAM, and RAM containing the channel enable registers (CHNENBLn) have unpredictable contents.

The active register set is assigned to channel 0 and the PC is initialized to all zeros. However, since the channel enable register is all zeros, there are no active channels and the SDMA is halted waiting for the boot channel to start.

The ARM platform will have to setup the SDMA in order to boot it. The CONFIG register must be initialized to determine the DMA/core clock ratio (1 or 2). Channel Enable Registers must also be initialized.

To start up the SDMA, the ARM platform first creates some channel control blocks (CCB) and buffer descriptors (BD) in ARM platform memory for the boot channel (channel 0) and then initializes the channel 0 pointer register (SDMA_MC0PTR) to the address of the first control block. Data Structures for Boot Code and Channel Scripts provides an overview of the data structure for the CCB and BD's. The SDMA_HSTART, SDMA_HOSTOVR and SDMA_EVTOVR registers are then configured according to Runnable Channels Evaluation to allow channel 0 to run.

Upon being enabled, the SDMA begins executing the script located at the address indicated by the Channel 0 Boot Address register (SDMA_CHN0ADDR) in the program memory. The reset value of SDMA_CHN0ADDR points to the default bootload script in ROM. This ROM script will read the channel 0 pointer register (SDMA_MC0PTR) to determine the location of the Channel Control Block (SDMA_CCB) in ARM platform memory. The script will then begin fetching by DMA the first channel control block which contains a pointer to the location channel 0 Buffer Descriptor chain which is also fetched via DMA. If the buffer descriptor contains a valid command, the script interprets the command in each buffer descriptor and proceeds to implement the command and move on to the next buffer descriptor control block. The buffer descriptor commands for

channel zero are typically set up to load SDMA's program RAM, Data RAM, and initial values for the channel contexts. Some channel scripts expect particular parameters to be passed

There are two ways to make the SDMA boot on a user-defined script. The OnCE (either via its JTAG interface or its ARM platform Control interface) can be used to download any code in the SDMA RAM and force the SDMA to boot on that code. Also, the SDMA_CHN0ADDR register in the ARM platform programming model can be modified to point to user code in RAM which would need to either have been loaded via the ONCE or default bootload routine (ex before a S/W reset).

55.4.8.2 Channel Script Execution

The execution of an SDMA script depends on both the instructions that make up the script, the data context upon which it operates, and commands or parameters allowed to the buffer. All these items must be initialized before the script is allowed to execute.

Each of the 32 channels has a separate context, but may share scripts and locations in data RAM.

Channels are initialized by the ARM platform by using channel 0 to download any required scripts and data values and the channels initial context. The context contains all the initial values of the SDMA core registers. This includes the Program Counter (PC) which is set to the start of the desired script in SDMA program memory.

The ARM platform selects which trigger conditions that must occur for the channel to start by configuring the SDMA_CHNENBL, SDMA_HOSTOVR and SDMA_EVTOVR registers. The trigger events include ARM platform setting HE (SDMA_HSTART) or a hardware DMA request asserts an event input to SDMA. The channel can become active according to its priority compared with other runnable channels when the selected trigger(s) cause the condition described in Runnable Channels Evaluation to evaluate as true.

The specific parameters to be passed to each script in the buffer descriptor or context are documented in the software documentation for each script. Please refer to SDMA Scripts for complete script documentation. Buffer Descriptor Format provides an overview of the buffer descriptor format.

55.4.8.3 Initialization and Script Execution Setup Sequence

To summarize, the following steps are minimally required to setup SDMA and run channel scripts.

- Perform Hardware Reset. The program RAM, context RAM, data RAM and SDMA_CHNENBLn registers have unpredictable contents after this reset.
- Initialize SDMA_CHNENBLn registers to map DMA request events to desired channels.
- Configure SDMA_CHNPRIn registers to select priority for runnable channels. A non-zero priority is required for the channel to run.
- Configure the SDMA_CONFIG register to select DMA to SDMA core clock ratio .
- Set up channel control blocks and buffer descriptors in ARM platform to specify the loading of SDMA program RAM and channel contexts for each SDMA channel to be used. Reference Data Structures for Boot Code and Channel Scripts.
- Configure SDMA_MC0PTR register with base address of ARM platform Channel Control Block base address.
- Initialize SDMA_CHNENBLn registers to map DMA request events to associated channel. Reference Mapping DMA Requests to Pending Channels.
- Configure SDMA_CHNPRIn registers to set priority for each channel to be run.
- For each channel to be run, configure SDMA_HOSTOVR (HO) and SDMA_EVTOVR (EO) registers to select which events (hardware and/or software trigger events) must occur for the channel to be runnable. Reference Runnable Channels Evaluation.
- Set bit 0 of the SDMA_HSTART register to set HE[0] and allow Channel 0 to run (assumes EO[0] andDO[0] were both set in previous step). This will cause SDMA to load the program RAM and channel contexts configured previously.
- Wait for Channel 0 to finish running. This is indicated by HI[0]=1 in the SDMA_SDMA_INTR register, or by optional interrupt to the ARM platform.
- Set the LOCK bit in the SDMA_SDMA_LOCK register to prevent un-authorized uploads of data to SDMA RAM.
- Additional channel scripts can now be run by enabling the selected software or hardware trigger event according to Runnable Channels Evaluation.

55.4.9 SDMA Programming Model

This section describes the programming model for the SDMA RISC engine, including its processor, memory, and internal control registers.

All addresses are related to the internal SDMA memory map, which is completely different from the ARM platform memory maps. The ARM platform processor has no access to any hardware resource described, except when those resources are described in ARM Platform Memory Map and Control Register Summary. .

55.4.9.1 State and Registers Per Channel

The SDMA can be seen as a set of 32 identical devices that are able to perform one data transfer channel each. Only one channel can work at a time, but every channel state is available at any time.

This chapter lists the components of every channel state.

55.4.9.2 General Purpose Registers

Each channel has eight general purpose registers of 32 bits for use by scripts. General register 0 has a dedicated function for the loop instruction, but otherwise can be used for any purpose.

55.4.9.3 Functional Unit State

Each channel context has some state that is part of the functional units.

The specific allocation of this state is part of the functional unit definition that is described in Burst DMA Unit Programming, Peripheral DMA Unit Programming .

This state must be saved/restored on context switches.

55.4.9.3.1 Program Counter Register (PC)

The PC is 14 bits. Since instructions are 16 bits in width and all memory in the SDMA is 32 bits in width, the low order bit of the PC selects which half of the 32-bit word contains the current instruction.

A low order bit of zero selects the most significant half of the word.[1]

55.4.9.3.2 Flags

Each channel has the following four flags:

- The T bit reflects the status of some arithmetic and test instructions. It is set when the result of an addition or a subtraction is zero and cleared otherwise. It is also the copy of the tested bits. Finally, it can also be set when the loop counter (GReg0) reaches zero. When the last instruction of the hardware loop is an operation that can modify the T flag, its effect on T is discarded and replaced by the GReg0 status.

1. For example, big-Endian.

- Two additional bits, SF and DF, are used to indicate error conditions resulting from loading data sources and storing to destinations, respectively. Access errors set these bits, and successful transactions clear them. They can also be cleared by specific instructions (CLRF and loop). The source fault (SF) is updated by the loads LD and LDF; the destination fault (DF) is updated by the stores ST and STF.
- Access errors are caused by several conditions including writing to the ROM, writing to a read-only memory mapped register, accessing an unmapped address, or any transfer error received by a peripheral when it is accessed.

  The SF and DF flags have a major impact on the behavior of the hardware loop: If SF or DF is set when starting a hardware loop and it is not masked by the loop instruction, the loop body will not be executed. Inside the loop body, if a load or store sets the corresponding SF or DF flag, the loop exits immediately. Testing the status of the T flag at the end of the loop (as well as testing both SF and DF) tells if the loop exited abnormally as any anticipated exit prevents GReg0 from reaching the zero value and thus setting the T flag. This is also valid if the fault occurs at the last instruction of the last loop.

- The last flag is the loop mode flag, LM, which is composed of two bits. The most significant bit indicates when the processor is currently operating in loop mode. It is set by the loop instruction and is cleared after execution of the last instruction of the last loop. The least significant bit is set when the program counter points to the last instruction of a loop on the last path. It is used for a channel that is restored with this configuration to know that the next program counter is EPC. As with the dynamic context switch Greg0, which indicates when the program must get out of the loop, it can be restored only on the last instruction of the loop. This, however, is too late to fetch the next instruction after the loop.

55.4.9.3.3 Return Program Counter (RPC)

The RPC is 14 bits. It is set by the jump to the subroutine instructions and used by the return from the subroutine instructions.

Instructions are available to transfer its contents to and from a general register.

55.4.9.3.4 Loop Mode Start Program Counter (SPC)

The SPC is 14 bits. It is set by the loop instruction to the location immediately following it.

55.4.9.3.5 Loop Mode End Program Counter (EPC)

The EPC is 14 bits. It is set by the loop instruction to the location of the next instruction after the loop.

55.4.9.4 Context Switching-Programming

Each channel has a separate context consisting of the eight general purpose registers and additional registers representing the state of the functional units.

The active registers and functional units contain the context of the active channel. The context of inactive channels is stored in SDMA RAM, which is part of the SDMA address space.

In a function of the selected context switching mode (Context Switching), modified registers by the program can be saved in the channel RAM space while the program is going on. In every cycle, a write access to the RAM is possible.

On a done or yield(ge) instruction, SDMA goes into "real" context switching. In one of the dynamic modes, modified registers not previously saved, as well as the PC-Loop registers, are stored into the context area of the channel that will be closed. The new PC-Loop registers are loaded from the context area of the new channel. All other registers are restored while the program is executed, giving priority to registers used by the decoded instruction. Therefore, in the best case, only the PC and Loop registers should be saved and restored during this context-switching phase, which only requires five SDMA cycles.

In static mode, the context switch stores all registers in the old channel RAM space, and restores all registers from the new channel RAM space. It requires 26 SDMA cycles.

The address of the context memory for channel *i* is *CONTEXT_BASE* + 24**i* or *CONTEXT_BASE* + *32*i w*here CONTEXT_BASE equals 0x0800. The table below presents the layout of a channel context in memory:

Table 55-11. Layout of a Channel Context in Memory for SDMA

OFFSET	31	30	29-16	15	14	13-0
0	SF	-	RPC	T	-	PC
1	LM		EPC	DF	-	SPC
2	GR0					
3	GR1					
4	GR2					
5	GR3					
6	GR4					
7	GR5					

Table continues on the next page...

Table 55-11. Layout of a Channel Context in Memory for SDMA (continued)

8	GR6
9	GR7
10	MDA (burst DMA)
11	MSA (burst DMA)
12	MS (burst DMA)
13	MD (burst DMA)
14	PDA (peripheral DMA)
15	PSA (peripheral DMA)
16	PS (peripheral DMA)
17	PD (peripheral DMA)
18	
19	
20	Reserved[1]
21	Reserved[1]
22	Reserved[1]
23	Reserved[1]
24	Scratch RAM (optional)
25	Scratch RAM (optional)
26	Scratch RAM (optional)
27	Scratch RAM (optional)
28	Scratch RAM (optional)
29	Scratch RAM (optional)
30	Scratch RAM (optional)
31	Scratch RAM (optional)

55.4.9.5 Address Space

The SDMA has four internal buses which are listed here.

- The Instruction bus reads instructions from the memory. Its address map is described in Instruction Memory Map.
- The Data bus (DMBUS) accesses the same memories as those visible on the Instruction bus, some memory-mapped registers (scheduler status and OnCE registers), and up to 14 peripherals. Its address map is described in Data Memory Map.

- The Functional Units bus (FUBUS) accesses the , Burst DMA, Peripheral DMA . The addressing mechanism is further detailed in Functional Units Programming Model.
- The Context Switch bus reads/writes registers into context-switch RAM space. It is a 64-bit bus dedicated for accessing this RAM space for updating the context of the running channel. While the program is going on, this bus has the lowest priority compared to the Instruction and Data buses, except for restoring a register needed for the decoded instruction to be executed. On the save part of a context switch (when the PCU is in its slave state), this is the only one used. On the restore part, the Instruction bus has the priority to read the next instruction at the restored PC and otherwise the Context Switch bus is used. It is not possible to control the actual data transfers that occur on this bus.

55.4.9.5.1 Instruction Memory Map

The instruction memory map is based on a 14-bit address bus and a 16-bit data (instruction) bus. Each address corresponds to a 16-bit data location.

Instructions are fetched from either program ROM or program RAM. An SDMA script is able to change the contents of the program RAM, which is also visible from the data bus.

The first two instruction locations (at 0 and 1) are special. Location 0 is where the PC is set on reset. Location 1 is where the PC is set upon the execution of an illegal instruction. It is expected that both of these locations will contain a jmp to handle routines.

Table 55-12. SDMA Instruction Memory Space

Device	SDMA Address (Hex)	Base Address Label	Block Name	WS	Description
ROM	0x0000 ↓ 0x07FF	SDMA_IBUS_ROM_ADDR	-	0	4 Kbyte internal ROM with boot code and standard routines.
RAM	0x1000 ↓ 0x1FFF	SDMA_IBUS_RAM_ADDR	-	0	8 Kbyte internal RAM with channels context and user data/routines.

55.4.9.5.2 Data Memory Map

All of the data accessible to SDMA scripts make up the data memory space of the SDMA.

This address space has several components:

- ROM (also visible on the Instruction bus)
- RAM (also visible on the Instruction bus)

- Shared Peripherals Registers
- SDMA Internal Registers (scheduler, OnCE, and registers that are also accessible by the ARM platform)

SDMA scripts can read and write to the context RAM, data RAM, shared peripheral registers, and internal registers.

The address range is 16 bits and the data width is 32 bits. Each address corresponds to a 32-bit data word. When accessing peripheral registers (USB and so on), the data width may be different. The exact address map for the peripherals depends on the project (as presented in each respective chapter).

Data access is performed with *ld* and *st* instructions that take the address from a general purpose register in the core (GReg*n*). The mapping between the general purpose register contents and the address bus is given in the following table:

Table 55-13. GReg*n* to DMBUS Address Mapping

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
sz	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
address															

Grayed bits are simply discarded but they must be cleared to ensure forward-script compatibility.

- sz (bit 31) indicates the peripheral data width: 0 is used for a 32-bit peripheral and 1 is used for a 16-bit peripheral.
- address (bits 15 down to 0) is the address of the accessed resource (internal memory, internal register, or shared peripheral).

Table 55-14. SDMA Data Memory Space

Device	SDMA Address (Hex)	Size	Description
ROM	0x0000 → 0x03FF	4 Kbyte	4 Kbyte internal ROM with boot code and standard routines
Reserved	0x0400 → 0x07FF	4 Kbyte	4 Kbyte Reserved
RAM	0x0800 → 0x0FFF	8 Kbyte	8 Kbyte internal RAM with channels contexts and user data/routines
per1	0x1000 → 0x1FFF	16 Kbyte	*peripheral 1* memory space (4 Kbyte peripheral's address space)
per2	0x2000 → 0x2FFF	16 Kbyte	*peripheral 2* memory space (4 Kbyte peripheral's address space)
per3	0x3000 → 0x3FFF	16 Kbyte	*peripheral 3* memory space (4 Kbyte peripheral's address space)
per4	0x4000 → 0x4FFF	16 Kbyte	*peripheral 4* memory space (4 Kbyte peripheral's address space)
per5	0x5000 → 0x5FFF	16 Kbyte	*peripheral 5* memory space (4 Kbyte peripheral's address space)
per6	0x6000 → 0x6FFF	16 Kbyte	*peripheral 6* memory space (4 Kbyte peripheral's address space)
Registers	0x7000 → 0x7FFF	16 Kbyte	Memory mapped registers

Table continues on the next page...

Table 55-14. SDMA Data Memory Space (continued)

Device	SDMA Address (Hex)	Size	Description
per7	0x8000 → 0x8FFF	16 Kbyte	*peripheral 7* memory space (4 Kbyte peripheral's address space)
per8	0x9000 → 0x9FFF	16 Kbyte	*peripheral 8* memory space (4 Kbyte peripheral's address space)
per9	0xA000 → 0xAFFF	16 Kbyte	*peripheral 9* memory space (4 Kbyte peripheral's address space)
per10	0xB000 → 0xBFFF	16 Kbyte	*peripheral 10* memory space (4 Kbyte peripheral's address space)
per11	0xC000 → 0xCFFF	16 Kbyte	*peripheral 11* memory space (4 Kbyte peripheral's address space)
per12	0xD000 → 0xDFFF	16 Kbyte	*peripheral 12* memory space (4 Kbyte peripheral's address space)
per13	0xE000 → 0xEFFF	16 Kbyte	*peripheral 13* memory space (4 Kbyte peripheral's address space)
per14	0xF000 → 0xFFFF	16 Kbyte	*peripheral 14* memory space (4 Kbyte peripheral's address space)

55.4.10 SDMA Initialization

Appendix A describes the setup of the SDMA . This section provides a quick description of several initialization procedures.

NOTE

There may be differences with the actual implementation in the API.

55.4.10.1 Hardware Reset-SDMA

After reset, the RAM that holds contexts, data, scripts, and the DMA request-channels matrix has unpredictable content.

The core registers are all reset to 0, including the PC; the PCU state is *Sleep after Reset*. No channel can be activated because all of the priorities are also reset to 0.

55.4.10.2 Standard Boot Sequence

The following is the standard boot sequence:

1. Initialize the CONFIG register-detailed in Configuration Register (SDMAARM_CONFIG)-to determine the ARM platform DMA/core clock ratio (1 or 2)
2. Initialize the DMA request-channels matrix (seeChannel Enable RAM (SDMAARM_CHNENBL*n*)).
3. Program the channel control registers-Channel Event Override (SDMAARM_EVTOVR), Channel BP Override (SDMAARM_DSPOVR), Channel

BP Override (SDMA_HOSTOVR), and Channel Event Pending (SDMAARM_EVTPEND)-according to the channel allocation.
4. Perform any necessary setup as required by the standard boot script in ROM (this is described in Appendix A).
5. Trigger channel 0 with the Channel Start (SDMAARM_HSTART) register, which starts the execution of the ROM script starting at address 0. This boot downloads channel scripts and contexts in RAM.

55.4.10.3 User-Defined Boot Sequence

The following is a user-defined boot sequence:

1. Initialize the Configuration Register (SDMAARM_CONFIG)Channel Enable RAM (SDMAARM_CHNENBL*n*), Channel Event Override (SDMAARM_EVTOVR), Channel BP Override (SDMAARM_DSPOVR), Channel ARM platform Override (SDMAARM_HOSTOVR), and Channel Event Pending (SDMAARM_EVTPEND).
2. Use the OnCE (either via its JTAG interface or its ARM platform control registers) to download any code in the SDMA RAM. Accessing the Memory describes how to write data to the RAM via the OnCE.
3. Use the OnCE instructions to make the PC default value point to the new boot script start address, or rely on the ROM startup script, which first jumps to the address in Channel 0 Boot Address (SDMAARM_CHN0ADDR). (This register default address points to the standard boot script.)

55.4.10.4 Script Loading and Context Initialization

The execution of an SDMA script depends on both the instructions that make up the script and the data context upon which it operates. Both must be initialized before the script is allowed to execute.

Each of the 32 channels has a separate data context, but may share scripts and locations in the data RAM.

The ARM platform manages the space in program RAM and data RAM. It also manages the assignment of SDMA channels to the device drivers that need them. Channels are initialized by the ARM platform via the channel 0 boot script. The boot channel downloads any required scripts with their data and the channels' initial contexts. Every context contains all the initial values of the registers, including the PC. Then the ARM platform can enable any channel that becomes active and begins fetching and executing instructions from its script.

55.4.11 Instruction Description

The following sections introduce the instruction of the SDMA.

Instruction set details are available in Instruction Set.

55.4.11.1 Scheduling Instructions

The following are scheduling instructions:

- done-The instruction causes certain scheduling or interrupt bits to be set or cleared, which may cause a change in the schedule-ability of the running channel. Then the instruction causes the SDMA to evaluate the current scheduling priorities and to choose the highest priority ready channel. If this channel is not the current channel, a context switch will take place. If there are no runnable channels, the SDMA will enter the stopped mode. The done 5 has a special usage reserved for debug, as explained in Debug Instructions.
- yield-These instructions are special cases of the done instruction. They do not modify the scheduling bits, but allow the highest pending channel (if it exists) to preempt the current channel if the pending channel priority is strictly greater than the current channel priority.
- yieldge-These instructions are special cases of the done instruction. They do not modify the scheduling bits, but allow the highest pending channel (if it exists) to preempt the current channel if the pending channel priority is strictly greater or equal to the current channel priority.
- notify-The notify instruction affects the scheduling bits, but does not cause rescheduling.

55.4.11.2 Conditional Branch Instructions

The conditional branch instructions of an 8-bit displacement, which is sign-extended and added to the current PC (which points to the next instruction) if the condition is satisfied.

Otherwise, control passes to the next sequential instruction.

- BF-Branch if False. The branch is taken if the T bit in the processor status is zero (false).
- BT-Branch if True. The branch is taken if the T bit in the processor status is one (true).

- BSF-Branch if Source Fault. The branch is taken if the SF bit in the processor status is one.
- BDF-Branch if Destination Fault. The branch is taken if the DF bit in the processor status is one.

55.4.11.3 Unconditional Jump Instructions

There are two varieties of unconditional control transfers: an absolute transfer and a through-register transfer.

Absolute transfers have a 14-bit address field that replaces the current PC.

- JMP-Jump. Causes the processor to jump to an absolute address encoded in the instruction itself.
- JSR-Jump to Subroutine. Causes the processor to jump to a subroutine, the address of which is encoded in the instruction itself.
- JMPR-Jump through Register. Causes the processor to jump to an absolute address contained in a General register. This instruction is meant to be used when more than one level of subroutines are required.
- JSRR-Jump to Subroutine through Register. Causes the processor to jump to a subroutine, the address of which is contained in a General register. This instruction is meant to be used when more than one level of subroutines are required.

55.4.11.4 Subroutine Return Instructions

The following are subroutine return instructions:

- RET-Return from Subroutine. The RET restores the contents of RPC to PC.
- LDRPC-Load from RPC to Register. THe LDRPC instruction is meant to be used when more than one level of subroutines are required. It stores the contents of RPC in any General register.

55.4.11.5 Loop Instruction

The following is a loop instruction:

LOOP-Enters Loop Mode. Before entering loop mode, the loop instruction can optionally clear the fault flags (SF and/or DF) based on a 2-bit field in the instruction. This feature is linked to the fact that setting SF or DF in loop mode will cause an immediate exit of the loop.

55.4.11.6 Miscellaneous Instructions

The following are miscellaneous instructions:

- CLRF-Clear Fault Flags. This instruction clears any combination of SF and DF.
- MOV r,s-This moves data from GReg[s] to GReg[r].
- LDI r,immediate-This loads GReg[r] with a zero-extended immediate value.

55.4.11.7 Logic Instructions

The following are logic instructions:

- XORr,s-This performs an exclusive or between GReg[r] and GReg[s], and stores the result in GReg[r].
- XORIr,immediate-This performs an exclusive or between GReg[r] and a zero-extended immediate value, and stores the result in GReg[r].
- ORr,s-This performs an or between GReg[r] and GReg[s], and stores the result in GReg[r].
- ORIr,immediate-This performs an or between GReg[r] and a zero-extended immediate value and, stores the result in GReg[r].
- ANDNr,s-This performs an and between GReg[r] and the negated GReg[s], and stores the result in GReg[r].
- ANDNIr,immediate-This performs an and between GReg[r] and the negated zero-extended immediate value, and stores the result in GReg[r].
- ANDr,s-This performs an and between GReg[r] and GReg[s], and stores the result in GReg[r].
- ANDIr,immediate-This performs an and between GReg[r] and a zero-extended immediate value, and stores the result in GReg[r].

55.4.11.8 Arithmetic Instructions

Arithmetic instructions modify the T bit in the processor status according to the result of the operation. The T bit is set if the result is zero, otherwise it is cleared.

- ADD r,s-This performs the addition of GReg[r] and GReg[s], and stores the result in GReg[r].
- ADDI r,immediate-This performs the addition of GReg[r] and a zero-extended immediate value, and stores the result in GReg[r].

- SUB r,s-This performs the subtraction of GReg[s] from GReg[r], and stores the result in GReg[r].
- SUBIr,immediate-This performs the subtraction of a zero-extended immediate value from GReg[r], and stores the result in GReg[r].

55.4.11.9 Compare Instructions

Compare instructions modify the T bit in the processor status according to the result of the operation. The T bit is set if the comparison is true, otherwise it is cleared.

NOTE

Only one version of the immediate form is implemented. Non-equality comparisons to immediate values will require two instructions.

- CMPEQ r,s-This sets T when registers GReg[r] and GReg[s] are equal.
- CMPEQIr,immediate-This sets T when register GReg[r] and the zero-extended immediate value are equal.
- CMPLTr,s-This sets T when register GReg[r] is less than and not equal to GReg[s]. The comparison is signed.
- CMPHS r,s-This sets T when register GReg[r] is greater than or equal to GReg[s]. The comparison is signed.

55.4.11.10 Test Instructions

Test instructions modify the T bit in the processor status according to the result of the operation. The T bit is set if any bit in the result is one, otherwise it is cleared.

- TSTr,s-This performs an and between GReg[r] and GReg[s], and sets T if the result is not zero.
- TSTIr,immediate-This performs an and between GReg[r] and a zero-extended immediate value, and sets T if the result is not zero.

55.4.11.11 Byte Permutation Instructions

These instructions shuffle the bytes in a register. For the purpose of describing these instructions, have the bytes in a register be numbered from the most significant as b_3, b_2, b_1, b_0.

- RORBr-The rotate right byte. The result is b_0, b_3, b_2, b_1.

- REVBr-The reverse bytes in word. The result is b_0, b_1, b_2, b_3.
- REVBLOr-The reverse, two low-order bytes. The result is b_3, b_2, b_0, b_1.

55.4.11.12 Bit Shift Instructions

The following are bit shift instructions:

- ROR1r-The rotate right 1 bit. This instruction does a circular right shift of 1 bit.
- LSR1r-The logical shift right 1 bit. This instruction shifts all bits to the right by 1. The high order bit is replaced by a 0.
- ASR1r-The arithmetic shift right 1 bit. This instruction shifts all bits to the right by 1. The high order bit is replaced by itself.
- LSL1r-The logical shift left 1 bit. This instruction shifts all bits to the left by 1. The low order bit is replaced by zero.

55.4.11.13 Bit Manipulation Instructions

- BCLRIr,n-The bit clear is immediate; clears bit number *i* in register *r*.
- BSETIr,n-The bit set is immediate; sets bit number *i* in register *r*.
- BTSTIr,n-The bit test is immediate; tests bit number *i* in register *r* (T becomes equal to the selected register bit).

55.4.11.14 SDMA Memory Access Instructions

All memory accesses are 32 bits.

Any memory location that is implemented with less than 32 bits (for example, peripheral registers) causes unimplemented bits to be read as 0s.

All memory accesses will cause either the SF or DF flags in the processor status to be set if they cause a fault.

What constitutes a fault, especially when accessing peripheral registers, is a property of the memory location.

- LDr,(b,d)-The load instruction creates an address by adding the displacement field (d) to the contents of the base register (b). The SDMA location at the resulting address is read and placed in the destination register (r).
- STr,(b,d)-The store instruction creates an address in the same manner as the load instruction. The register (r) is stored in the SDMA location at the resulting address.

55.4.11.15 Functional Unit Instructions

The functional unit instructions have an 8-bit field that is placed on the functional unit bus.

Some of these bits are used to select which functional unit should be involved in the transfer. The remaining bits are decoded by the selected functional unit so their specific use depends on the functional unit. See Functional Units Programming Model.

There are two functional unit instructions, as follows:

- LDFr,fub-The 8-bit field is placed on the functional unit bus and a read is issued to the selected functional unit. As a result of this instruction, the SF may be set in the processor status.
- STFr,fub-The 8-bit field is placed on the functional unit bus and a write is issued to the selected functional unit. As a result of this instruction, the DF may be set in the processor status.

55.4.11.16 Illegal Instructions

All instruction encodings that are illegal cause the following actions:

- The current PC (which points to one beyond the offending instruction) is put in the EPC register.
- The loop mode bit is cleared.
- The PC is set to the value stored in the Illegal Instruction Trap Address (SDMAARM_ILLINSTADDR) register (the default value is 0x0001).

ILLEGAL-Although any instruction other than those indicated in the SDMA specification will trigger the illegal instruction mechanism, the ILLEGAL instruction code is preferred as it will always be kept as *illegal* in the possible future versions of the SDMA core.

55.4.11.17 Debug Instructions

The following are debug instructions:

- SOFTBKPT-The software breakpoint instruction causes the core to stop and enter debug mode. The core can then be accessed and started by the OnCE debug block only.

- done 5-This instruction is used for debugging, as it copies the contents of the PCU registers and flags to the context memory. Information on this instruction is described in Saving the Context.
- CpShReg-This instruction copies the context memory into the PCU registers and flags. Modifying the corresponding memory location before executing this instruction enables you to have the channel continue from a new instruction address. This instruction is described in Restoring the Context.

55.4.12 Functional Units Programming Model

The functional unit instructions cause an 8-bit code, found in the low eight bits of the instruction, to be asserted on the functional unit control bus.

Some of these bits are used to select one of several functional units. Functional units which can be selected include SDMA registers such as MSA and MSD which are not mapped in the SDMA memory map, and are accessible only through the functional unit bus. These Functional Unit Registers are listed in the following table. In order to establish a programming convention, assume the selection bits are some number of the most significant bits of the 8-bit code. Furthermore, some number of the least significant bits is decoded by a given functional unit to establish the type of operation to perform.

Table 55-15. Functional Unit Registers

Functional Unit	Register	Register Name	Section/Page
Burst DMA Unit Programming	SDMSA	Memory Source Address Register	Memory Source Address Register (MSA)
	MDA	Memory Destination Address Register	Memory Destination Address Register (MDA)
	MD	Memory Data Buffer Register	Memory Data Buffer Register (MD) (Write) Burst DMA Write (stf) (Read) Burst DMA Read (ldf)
	MS	Memory State Register	State Register (MS)
Peripheral DMA Unit Programming	PSA	Peripheral Source Address Register	Peripheral Source Address Register (PSA)
	PDA	Peripheral Destination Address Register	Peripheral Destination Address Register (PDA)
	PD	Peripheral Data Buffer Register	Peripheral Data Register (PD) (Write) Peripheral DMA Write (stf)-Write Mode

Table continues on the next page...

Table 55-15. Functional Unit Registers (continued)

Functional Unit	Register	Register Name	Section/Page
			(Read) Peripheral DMA Read (ldf)-Read Mode
	PS	Peripheral State Register	Peripheral State Register (PS)

More information regarding the functional units can be found in Peripheral DMA Unit, and Burst DMA Unit.

55.4.12.1 Burst DMA Unit Programming

The DMA instructions control the DMA state machine and may cause a DMA cycle on the associated memory bus.

There are four registers associated with the burst DMA unit: a Memory Source Address register (MSA), a Memory Destination Address register (MDA), a Memory Data buffer (MD), and a state register (MS). The burst DMA has two different uses:

- A data transfer between External Memory Interface and SDMA general register
- A data transfer in copy mode where blocks of data are transferred from the source address to the destination address

55.4.12.1.1 Memory Source Address Register (MSA)

The source address register contains the pointer into EXTMC memory associated with the next read data transfer. It has byte granularity.

Reading the register with the ldf instruction has no side effects, and gives the address value in the EXTMC memory of the next data that is read by the SDMA during an ldf MD instruction.

Writing the source address register has two side effects: If the prefetch bit is set, a DMA read cycle (8-word read access) is issued with the new address. Any data still located in the buffer is lost. If there is valid write data in the buffer, it is necessary to force the DMA to completely flush it out before modifying MSA to guarantee all the data is effectively written to memory.

The MSA register has two modes of programming:

- Frozen-In frozen mode, the MSA register is not modified after DMA accesses.
- Incremented (default mode)-In incremental mode, MSA is incremented by the number of bytes transferred during read cycles.

55.4.12.1.2 Memory Destination Address Register (MDA)

The destination address register contains the pointer into EXTMC memory associated with the next write data transfer. It has byte granularity.

Reading the MDA register with the ldf instruction has no side effects. It gives the address value in the EXTMC memory where the next SDMA data (stf r,MD instruction) is stored when MD FIFO is flushed.

Writing the destination address register has one side effect. Any data still located in the buffer is lost. If there is valid write data in the buffer, it is necessary to force the DMA to completely flush it out before modifying MDA to guarantee all the data is effectively written to memory.

The MDA register has two modes of programming:

- Frozen-In frozen mode, the MDA register is not modified after DMA accesses.
- Incremented (default mode)-The MDA register is incremented by the number of bytes transferred during write cycles.

55.4.12.1.3 Memory Data Buffer Register (MD)

The data buffer register consists of a bank of 36 bytes that behave like FIFO.

This FIFO stores the eight words received when a read burst is triggered by the DMA (DMA is in read mode).

The MD register is in write mode after a writing in MDA or after an stf MD instruction.

In that case, a burst write access is automatically triggered when there are more than eight words in MD. For bandwidth optimization, any transfers between DMA and the EXTMC controller are based on burst accesses.

An ldf r,MD|SIZE instruction that reads the data buffer may cause a DMA cycle, as follows:

- If there are less bytes in the FIFO than the size parameter of the instruction. For instance, if only two bytes are available in MD and a 4-byte read is requested, a burst read access is executed to complete the two bytes.
- If the prefetch bit is set, and after reading there is enough space in the FIFO to store a full burst, a burst read access is triggered.

An stf r,MD|SIZE instruction that writes to the data buffer may cause a DMA cycle if the number of written bytes in MD is higher than 32 (eight words) or if the flush bit is set.

When DMA is used for data transfer between SDMA and EXTMC (reading or writing), no immediate error is possible because the block manages a data misalignment issue; therefore, it is allowed to read/write a word to/from a half-word address. However, the addresses (source or destination) must belong to the EXTMC memory mapping. The only potential error, in this mode, would be the error sent back by the EXTMC controller when an access to a super-user page is detected. The whole transfer on the DMA associated bus will be considered successful when there are no errors seen on the bus during the transfer. In copy mode, an immediate error could be returned to SDMA as described in Burst DMA Unit Error Management.

55.4.12.1.4 State Register (MS)

The state register contains the DMA state-machine value. It can be accessed in case of an error received during a transfer. MS is also accessed to set-up the conditional yielding feature.

The initialization value of this register is 0 and it consists of the following:

Table 55-16. SDMA_MS Structure

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	spriv	stype	0	0	dpriv	dtype
W																
R	0	0	0	0	y	d	e		0	0	n					
W																

Table 55-17. SDMA_MS Field Descriptions

Field	Description
31-22	Reserved
21 spriv	The spriv value is ignored for this device. 0 = valid value 1 = Reserved
20 stype	Source Mode. Indicates if MSA has to be incremented (or not) during accesses. 0 Frozen-MSA is not modified. 1 Incremented-MSA is incremented by the number of transferred bytes during read access.
19-18	Reserved
17 dpriv	The dpriv value is ignored for this device. 0 = valid value 1 = Reserved
16	Destination Mode. Indicates if MDA has to be incremented (or not) during accesses.

Table continues on the next page...

Table 55-17. SDMA_MS Field Descriptions (continued)

Field	Description
dtype	0 Frozen-MDA is not modified. 1 Incremented-MDA is incremented by the number of transferred bytes during write access.
15-12	Reserved
11 y	Conditional Yielding selector. When selected, theyield/yieldge instructions will not switch channels if the Burst DMA is in Write Mode, and it has less than four bytes in its FIFO. This is aimed at reducing the number of inefficient FIFO flushes due to context switches. 0 Always yields 1 Yields conditionally (when there are less than four bytes in the FIFO in write mode)
10 d	Access Direction or DMA Mode. DMA is in write mode when data was written into MD by stf MD instructions, or if a previous DMA cycle on the external bus was a write access. Writing MDA or MSA changes the DMA mode to the respective value. DMA is in read mode when a previous DMA cycle was a read access, and DMA stays in read mode when data is read by SDMA with an ldf MD instruction. Reading MDA or MSA does not change the DMA mode. 0 Read Mode 1 Write Mode
9-8 e	Error. Indicates if the previous access was acknowledged with a bus error. 00 No error was received. 01 *reserved* 10 Error mode 11 error read burst
7-6	Reserved
5-0 n	Number of bytes in the MD FIFO.

55.4.12.1.5 Burst DMA Write (stf)

When received from a stf instruction, the function code bits are interpreted as follows, depending on the addressed register:

Table 55-18. STF Code Bits

Register	7	6	5	4	3	2	1	0
MSA	s		p	freeze	r			spriv
MDA								dpriv
MD			f	cpy			sz	
MS								

Table 55-19. STF Code Bit Field Descriptions

Field	Description
7-6 s	Functional Unit selector 00 for Burst DMA
5 p (MSA)	Prefetch Flag 0 No prefetch 1 Prefetch required from new MSA
5 f (MD)	Forced Flush Flag 0 Automatic flush 1 FIFO contents are flushed (including the new written data).
4 freeze (MSA/MDA)	Address Freeze Mode 0 Address is normally incremented. 1 Address is frozen.
4 cpy (MD)	Copy Mode selection 0 Write Mode 1 Copy Mode
3-2 r	Register selection 00 MSA 01 MDA 10 MD 11 MS
1-0 sz (MD/MS)	Transfer Size 00 size 0 (no data stored in the FIFO) 01 byte (8 bits) 10 half-word (16 bits) 11 word (32 bits)
0 spriv (MSA)	The spriv value is ignored for this device. 0 = valid value 1 = Reserved
0 dpriv (MDA)	The dpriv value is ignored for this device. 0 = valid value 1 = Reserved

The possible write instructions are listed in the table below (unused bits should always be cleared).

Table 55-20. Burst DMA STF Instruction List

Binary	Assembly	Comments
00_0_0_00_00	stf r,MSA	Writes content of the SDMA general register (r) to the source address register. MSA is in incremented mode.

Table continues on the next page...

Table 55-20. Burst DMA STF Instruction List (continued)

Binary	Assembly	Comments
00_0_1_00_00	stf r,MSA|FR	Writes content of the SDMA general register (r) to the source address register. MSA is in frozen mode.
00_1_0_00_00	stf r,MSA|PF	Writes content of the SDMA general register (r) to the source address register, and starts a read burst access. MSA is in incremented mode.
00_1_1_00_00	stf r,MSA|PF|FR	Writes content of the SDMA general register (r) to the source address register, and starts a read burst access.
00_0_0_01_00	stf r,MDA	Writes content of the SDMA general register (r) to the destination address register. MDA is in incremented mode.
00_0_1_01_00	stf r,MDA|FR	Writes content of the SDMA general register (r) to the destination address register. MDA is in frozen mode.
00_1_0_10_00	stf r,MD|SZ0|FL	No data transfers between the SDMA and MD, but all valid written data of the MD is flushed to the memory. An acknowledge or error is sent back to the SDMA core on transfer completion.
00_0_0_10_01	stf r,MD|SZ8	8-bit (byte) transfer to write buffer MD
00_1_0_10_01	stf r,MD|SZ8|FL	8-bit (byte) transfer to write buffer MD and flush after transfer. All valid written data of the MD is flushed to memory.
00_0_0_10_10	stf r,MD|SZ16	16-bit (half-word) transfer to write buffer MD
00_1_0_10_10	stf r,MD|SZ16|FL	16-bit (half-word) transfer to write buffer MD and flush after transfer. All valid written data of the MD is flushed to memory.
00_0_0_10_11	stf r,MD|SZ32	32-bit (word) transfer to write buffer MD
00_1_0_10_11	stf r,MD|SZ32|FL	32-bit (word) transfer to write buffer MD and flush after transfer. All valid written data of MD is flushed to memory.
00_0_1_10_00	stf r,MD|CPY	No data transfer between SDMA and MD but starts a copy transfer whose length is given by the 4 LSB of *r* register. (Maximum burst length is eight words.)
00_0_0_11_11	stf r,MS	32-bit (word) transfer to status register MS
00_0_0_11_00	stf r,MS|SZ0	Clears the error flag (if set). Other MS bits are unchanged; this instruction is also known as clref MS.

NOTE

When a flush bit is set, the SDMA flushes the FIFO including the newly written data. An acknowledge is sent to the core before the flush completes (except if size 0 is used). The goal of this flush bit is to force a flush, but it is recommended to use it only when needed (for example, when finishing a row of pixels during 2D data transfers). Indeed, if this bit is omitted and if there are more than 32 bytes in the FIFO, a burst write access is automatically triggered.

Since all the stf r,MD instructions (including the copy mode) acknowledge the SDMA core before the store is effective (except if size 0 is used), it is recommended to perform an ldf

from MS before terminating a channel in order to check the final error status. (The ldf from MS will stall the core until all the data was flushed out and the transfer status is known.)

After every stf MD instruction, the MDA is incremented by the number of bytes that are written in MD, except when it is programmed in frozen mode.

55.4.12.1.6 Burst DMA Read (ldf)

When received from an ldf instruction, the function code bits are interpreted as follows, depending on the addressed register:

Table 55-21. LDF Code Bits

Register	7	6	5	4	3	2	1	0
MSA	s				r			
MDA								
MD			p				sz	
MS								

Table 55-22. LDF Code Bit Field Descriptions

Field	Description
7-6 s	Functional Unit selector 00 for Burst DMA
5 p (MD)	Prefetch Flag 0 no prefetch 1 automatic prefetch
3-2 r	Register selection 00 MSA 01 MDA 10 MD 11 MS
1-0 sz (MD)	Transfer Size 00 reserved 01 byte (8 bits) 10 half-word (16 bits) 11 word (32 bits)

The table below lists the possible write instructions (unused bits should always be cleared).

Table 55-23. Burst DMA LDF Instruction List

Binary	Assembly	Comments
00_0_0_00_00	ldf r,MSA	Copies the source address register value into an SDMA general register. It gives the memory address of the next data that will be read with an ldf MD instruction.
00_0_0_01_00	ldf r,MDA	Copies the destination address register value into an SDMA general register. It gives the memory address where the next incoming data will be flushed.
00_0_0_10_01	ldf r,MD|SZ8	8-bit (byte) read
00_1_0_10_01	ldf r,MD|SZ8|PF	8-bit (byte) read. If after this reading and the MD FIFO is empty, a burst read access at the MSA address is triggered.
00_0_0_10_10	ldf r,MD|SZ16	16-bit (half-word) read
00_1_0_10_10	ldf r,MD|SZ16|PF	16-bit (half-word) read. If after this reading, and the MD FIFO is empty, a burst read access at the MSA address is triggered.
00_0_0_10_11	ldf r,MD|SZ32	32-bit (word) read
00_1_0_10_11	ldf r,MD|SZ32|PF	32-bit (word) read. If after this reading and the MD FIFO is empty, a burst read access at the MSA address is triggered.
00_0_0_11_00	ldf r,MS	Copy the status register value into an SDMA general register.

NOTE

Read data is 0-extended before writing in the SDMA general registers. When reading the MD register, the DMA takes data from the FIFO if it is available. If part or whole data is not in the FIFO, an external burst read access is performed to provide the missing data. The SDMA is stalled as long as the required read data is not complete.

After every reading, MSA is incremented by the number of read bytes from MD FIFO, except when MSA is programmed in frozen mode.

55.4.12.1.7 Prefetch/Flush and Auto-Flush Management-Burst DMA Unit

The prefetch and auto-flush management enables the SDMA RISC machine to go on while a DMA access is performed.

When the RISC core requires a prefetch (p = 1) to the Burst DMA, it will receive an immediate transfer acknowledge before the DMA has finished the external access. This enables the RISC core to do other things like accessing another DMA machine.

The basic principle in prefetch mode is for the DMA to anticipate data reads from the SDMA RISC engine by fetching external bursts of data as soon as there is enough space in the DMA FIFO to store it. If ever the RISC engine required data that is not available in the FIFO, the read acknowledge is delayed until the data is available, but it does not have to wait until the burst completes.

The auto-flush basic principle is similar: An automatic flush is triggered every time there are eight words to be written in the FIFO. If the FIFO is full and the RISC engine requires another write, it is stalled until the burst has started and enough space was freed in the FIFO to store that new data. This means the SDMA RISC engine does not have to wait for the completion of a burst to receive its acknowledge and continue its processing.

In particular, an auto-flush is executed when DMA is in write mode and if the following is true:

- If the FIFO is empty and the first write is to a word-aligned address of any size (ex: the 2 LSB of MDA[1:0]= 0x0), the auto-flush is triggered immediately after the write of the 32'nd byte.
- If the FIFO is empty, and if MDA is an odd byte address (1, 3, 5, 7,...) and an stf MD|SZ8 is executed, the byte is flushed to memory. Once MDA increments to a word aligned address, the auto-flush will be triggered every 32 bytes.
- If the FIFO is empty, and if MDA is a half-word address (2, 6, 0xA,...) and an stf MD|SZ16 is executed, the two bytes of the incoming data are flushed to memory. Once MDA increments to a word aligned address, the auto-flush will be triggered every 32 bytes.
- If the FIFO is empty, and if MDA is not a word-aligned address (ex 1, 2, 3, 5, 6, 7, 9,...), and an stf MD|SZ32 is executed, the first 1 to 3 bytes will be flushed up to the next word aligned address. Afterwards, an auto-flush will be triggered each time the FIFO receives 32-bytes.
- Therefore, if an stf MD|SZ32 is executed with MDA equal to 0x1 and with an empty MD FIFO, the bytes located at addresses 1, 2, and 3 are flushed, and the byte located at address 4 remains in MD FIFO. This solves the misalignment issue. Additionally, the next write instructions (stf) complete the FIFO until it contains eight words; then a burst write is executed by the DMA to empty the FIFO. Protocol on the external bus does not support bursts of different data types (byte, half-word, or word).

  For example, consider the case where data is written using a byte access, stf MD|SZ8. The value of MDA during the very first byte write determines when the auto-flush will occur as follows:

- If MDA=0x0, the flush occurs following the write of byte 32
- If MDA=0x1, the flush occurs following the write of byte 1, byte 3 and byte 35.
- If MDA=0x2, the flush occurs following the write of byte 2 and byte 34.

- If MDA=0x3, the flush occurs following the write of byte 1 and byte 33.
- If MDA=0x4, the flush occurs following the write of byte 32

The flush command forces the DMA to flush all MD valid bytes to the EXTMC controller. An acknowledge is sent immediately to the SDMA, and any potential error is reported on a future access. It is thus essential to conclude a transfer with a last read from MS, which will stall the core until all data was flushed out and returned to the transfer status (acknowledge or error).

NOTE

During this kind of auto-flush (which occurs only at the beginning of a misaligned write transfer) no acknowledge is sent back to the SDMA, which is stalled until a flush is completed.

55.4.12.1.8 Data Alignment and Endianness-Burst DMA Unit

55.4.12.1.8.1 Burst DMA in Read Mode

For every read access to MD, the data returned to the SDMA core and the new FIFO state depends on the MSA status and the access size.

The FIFO is considered as a stack of 36 bytes: Data is fetched externally on a 32-bit bus, but the valid bytes only are stored in the FIFO and left-aligned (for a transfer of consecutive words, it is only the first word that may be truncated). The following table shows the FIFO byte alignment strategy and the corresponding MSA, the returned data, and the new FIFO state for any access size of an internal read from MD.

Table 55-24. FIFO Read Configuration

Before read		Internal read access size	Read data	After read	
MSA[1:0]	FIFO state			MSA[1:0]	FIFO state
00	x0 x1 x2 x3 y0 y1 y2 y3 z0 z1 z2 z3 and so on...	sz8	00 00 00 x0	01	x1 x2 x3 y0 y1 y2 y3 z0
		sz16	00 00 x0 x1	10	x2 x3 y0 y1 y2 y3 z0 z1
		sz32	x0 x1 x2 x3	00	y0 y1 y2 y3 z0 z1 z2 z3
01	x1 x2 x3 y0 y1 y2 y3 z0 z1 z2 z3 t0 and so on...	sz8	00 00 00 x1	10	x2 x3 y0 y1 y2 y3 z0 z1
		sz16	00 00 x1 x2	11	x3 y0 y1 y2 y3 z0 z1 z2
		sz32	x1 x2 x3 y0	01	y1 y2 y3 z0

Table continues on the next page...

Table 55-24. FIFO Read Configuration (continued)

Before read		Internal read access size	Read data	After read	
MSA[1:0]	FIFO state			MSA[1:0]	FIFO state
					z1 z2 z3 t0
10	x2 x3 y0 y1 y2 y3 z0 z1 z2 z3 t0 t1 and so on...	sz8	00 00 00 x2	11	x3 y0 y1 y2 y3 z0 z1 z2
		sz16	00 00 x2 x3	00	y0 y1 y2 y3 z0 z1 z2 z3
		sz32	x2 x3 y0 y1	10	y2 y3 z0 z1 z2 z3 t0 t1
11	x3 y0 y1 y2 y3 z0 z1 z2 z3 t0 t1 t2 and so on...	sz8	00 00 00 x3	00	y0 y1 y2 y3 z0 z1 z2 z3
		sz16	00 00 x3 y0	01	y1 y2 y3 z0 z1 z2 z3 t0
		sz32	x3 y0 y1 y2	11	y3 z0 z1 z2 z3 t0 t1 t2

55.4.12.1.8.2 Burst DMA in Write Mode

For every write access to the MD, the new FIFO state depends on the MDA status and the access size.

The FIFO is considered as a stack of 36 bytes: Data is stored in the FIFO according to the internal access size and the former MDA value. The following table shows the FIFO byte alignment strategy corresponding to MDA, as well as the new FIFO state for any access size of an internal write to MD.

Table 55-25. FIFO Write Configuration

Before write		Internal write access size	Written data	After write	
MDA[1:0]	FIFO state			MDA[1:0]	FIFO state
00	tt uu vv ww ?? ?? ?? ?? ?? ?? ?? ?? and so on...	sz8	?? ?? ?? x0	01	tt uu vv ww x0 ?? ?? ?? ?? ?? ?? ??
		sz16	?? ?? x0 x1	10	tt uu vv ww x0 x1 ?? ?? ?? ?? ?? ??
		sz32	x0 x1 x2 x3	00	tt uu vv ww x0 x1 x2 x3 ?? ?? ?? ??
01	tt uu vv ww	sz8	?? ?? ?? x0	10	tt uu vv ww

Table continues on the next page...

Table 55-25. FIFO Write Configuration (continued)

Before write		Internal write access size	Written data	After write	
MDA[1:0]	FIFO state			MDA[1:0]	FIFO state
	xx ?? ?? ?? ?? ?? ?? ?? and so on...				xx x0 ?? ?? ?? ?? ?? ??
		sz16	?? ?? x0 x1	11	tt uu vv ww xx x0 x1 ?? ?? ?? ?? ??
		sz32	x0 x1 x2 x3	01	tt uu vv ww xx x0 x1 x2 x3 ?? ?? ??
10	tt uu vv ww xx yy ?? ?? ?? ?? ?? ?? and so on...	sz8	?? ?? ?? x0	11	tt uu vv ww xx yy x0 ?? ?? ?? ?? ??
		sz16	?? ?? x0 x1	00	tt uu vv ww xx yy x0 x1 ?? ?? ?? ??
		sz32	x0 x1 x2 x3	10	tt uu vv ww xx yy x0 x1 x2 x3 ?? ??
11	tt uu vv ww xx yy zz ?? ?? ?? ?? ?? and so on...	sz8	?? ?? ?? x0	00	tt uu vv ww xx yy zz x0 ?? ?? ?? ??
		sz16	?? ?? x0 x1	01	tt uu vv ww xx yy zz x0 x1 ?? ?? ??
		sz32	x0 x1 x2 x3	11	tt uu vv ww xx yy zz x0 x1 x2 x3 ??

NOTE

If the FIFO mode changes from a write to a read mode, all remaining written bytes in MD are lost but no error is returned. Typically, this happens if an ldf MD is executed after stf MD instructions. Before a mode change, it is recommended to force the flush of a potential remaining byte by a stfMD|SZ0|FL instruction. In the same way, if a FIFO mode changes from a read to a write mode, all prefetched data present in the FIFO is lost and no error is returned.

55.4.12.1.8.3 Endianness-Burst DMA Unit

Big and Little Endian are supported by the Burst DMA, but data is always stored in MD in Big Endian.

Byte manipulation is performed when data is exchanged with an Burst controller (for example, during read or write burst accesses).

55.4.12.1.9 Burst DMA Unit Copy Mode

A mechanism is available to perform fast ARM-to-ARM transfers.

Data does not flow through the SDMA core: It is kept in the DMA FIFO. This mechanism is selected when writing MD with a special option in the instruction code (copy flag).

It is possible to transfer up to eight words in one SDMA instruction (this does not mean in one cycle). In this mode, every time an stf MD|CPY is executed, a read burst is executed and directly followed by a write burst transfer. Burst transfers are limited to eight words. The size of the transfer (in words)-given by the SDMA general register (4 LSB)-is also limited to eight. The following SDMA code shows how 100 bytes could be copied from the MSA address to the MDA address. This is sample code only.

Burst DMA copy mode example

```
          ldi r0,@src
          stf r0,MSA                          // Source address setup
          ldi r1,@dst
          stf r1,MSA                          // Destination address setup
          ldi r0,0x64                         // data transfer counter
          ldi r1,0x8
MAIN_XFER:
          cmphs r0,r1                         // Is r0 >= 0x8
          bf LAST_XFER                      // If not, jump to last transfer label
          stf r1,MD|CPY                        // Copy 8 words from MSA to MDA address.
          subi r0,0x8                         // Decrement counter
          jmp MAIN_XFER                     // return to main transfer loop
LAST_XFER:
          stf r0,MD|CPY
```

The main transfer loop is executed 12 times; then r0 equals 4 and the last transfer loop is run.

In this mode, an acknowledge is transmitted to the core as soon as the read burst can start; thus, a first copy instruction returns an immediate acknowledge and subsequent copy instructions will be acknowledged as soon as the previous copy has finished.

55.4.12.1.10 Burst DMA Unit Error Management

Another point to consider is the management of errors.

Because the DMA immediately sends an acknowledge to the RISC core (except for the stf MS|SZ0|FLS instruction), it assumes no error will occur. If an error occurs, it is flagged (transfer error acknowledge) for the following DMA access.

This should not be a problem if the DMA is used properly. The MD accesses are meant to stall the SDMA as little as possible to optimize throughput and hide calculation time. Therefore, final access to MS should be performed before closing a channel. This access waits until any pending operation is finished in the burst DMA and gather any remaining error.

In copy mode, an error could be immediately returned to the SDMA on execution of the ldf copy or stf copy instruction. It happens when MSA or MDA are not word addresses (for example, 0[4]). This is because copy mode must only be used for transferring a large packet of aligned data.

When an error is received during a *read* transfer to the external bus, which may occur during the burst accesses, the MD FIFO contains the valid beats of the burst, and the error flag of MS is set to 2'b11 (error read burst). It is possible to read MS ("n" field) to know how much valid data remains in MD and when MD is empty (after ldf instructions). The next read MD instruction sets the MS error flag to 2'b10 (error mode), and an error is sent back to the SDMA core. In error mode, it is possible to read MSA, which gives the address of the error data. Any attempt to read or write MD, or to modify MDA or MSA in error mode, gives rise to an error; therefore, an error flag must be reset by clearing MS at the end of the SDMA code section responsible for error management.

In "error read burst" mode, writing MDA, MSA, or MD, or starting a copy transfer by a stf MD|COPY instruction will cancel the error mode. The following table shows when an immediate error is sent back according to the executed instruction.

Table 55-26. Possibilities in ERROR READ BURST Mode

DMA Instruction	Immediate Error	Comments
stf rn, MD stf rn, MSA (|U |PF) stf rn, MDA stf rn,MD|COPY	NO	Error mode is reset. MSA, MDA, or MD are updated and a DMA cycle may start. For the stf MD|COPY, a copy loop is executed.
stf rn, MS	NO	MS is updated.

Table continues on the next page...

Table 55-26. Possibilities in ERROR READ BURST Mode (continued)

DMA Instruction	Immediate Error	Comments
ldf rn, MS ldf rn, MSA ldf rn, MDA	NO	MS, MSA, and MDA could be read in ERROR READ mode without any side effects (for example, no DMA cycle is triggered).
ldf rn, MD	YES/NO	Immediate error if there is no more data available for read in the FIFO.

When an error is received during a *write* transfer, the error is reported to the next DMA access. In this case, an error is sent to the SDMA core and the DMA goes to its error mode. Reading MS gives the number of bytes that remain in MD; reading MDA gives the address of the error data. Any attempt to read or write MD, or to modify MDA or MSA in error mode, give rise to an error; therefore, an error flag must be reset by clearing MS at the end of the SDMA code section responsible for error management.

Table 55-27. Possibilities in ERROR Mode

DMA Instruction	Immediate Error	Comments
stf rn, MD stf rn, MSA stf rn, MDA	Yes	Any attempt to modify MD, MSA, MDA will raise an immediate error and burst DMA remains in error mode. When address registers are write-accessed, an error is returned.
stf rn, MS	No	This is the only way to exit error mode. MS[9:8] must be reset by an stf MS|SZ0 instruction.
ldf rn, MS ldf rn, MSA ldf rn, MDA	No	MS, MSA, and MDA could be read in error mode without any side effects (for example, no DMA cycle is triggered).
ldf rn, MD	Yes	Whatever the DMA direction (read or write), an ldf rn triggers an immediate error.

55.4.12.1.11 Conditional Yielding-Burst DMA Unit

The standard SDMA transfer is based upon a hardware loop that has the following structure:

Hardware Loop

```
loop
load Rn,source            // can be ldf or ld
<computation>            // can be done through functional units
store Rn,dest              // can be st or stf
done 0               // yield
```

This structure needs to be kept independent of the functional units' particularities regarding the context switch. However, there can be variations in the context switch's efficiency, which can depend on the number of data received up to that point, and on the data itself.

The DMA, with its 8-word burst capability, has a preferable context switch period when its address register is 8-word aligned: It is the only moment that occurs once every eight loops when the succession of bursts is not broken by the context switch. When this is not the case, a context switch requires the storing (or loading) of less than eight words, which requires separate accesses and is far less efficient. The rest of the 8-word packet is stored (or loaded) after the context restore, and this is done as separate accesses.

The proposed solution is a conditional yielding, which occurs only when the DMA is in an optimum state. It does not require any modification to the scripts. The condition is decided at the DMA level.

The DMA can be programmed in two modes-conditional or always-true-for every channel, which provides complete flexibility. By default, the DMA is not in conditional mode.

The DMA condition is computed from the FIFO fill level and the various modes, as follows:

- When copy mode is selected, regardless of the transfer direction ('read' or 'write'), the condition is always true.
- In read mode, the condition is always true.
- In write mode, the condition is true when there are four bytes or less in the FIFO; it is false when there are more than four bytes. The 4-byte limit comes from the possibility of saving those bytes as MD with absolutely no impact on the bus accesses.

The aim at conditional yielding is to avoid splitting bus accesses (especially bursts).

55.4.12.2 Peripheral DMA Unit Programming

The peripheral DMA unit is connected to the Multi-Layer DMA Crossbar Switch of the ARM platform.

Its goal is to perform data transfers between any blocks connected to the DMA bus of this platform. These blocks are either peripherals or memories. The peripheral DMA could be seen as the ARM platform DMA controller.

The DMA performs data transfers in three modes:

- Read mode, where data is read from peripherals or from memory connected to the ARM platform and copied in a SDMA general register.
- Write mode, where data of a general register has to be written in a peripheral or a memory.
- Copy mode, where data is read from a peripheral (or memory) at a source address (PSA) and automatically written to a peripheral (or memory) at a destination address (PDA).

In copy mode, no SDMA general register is involved as transferred data only goes through the data register of the DMA.

The peripheral DMA has three addressing modes: frozen, incremented, and decremented, as follows:

- Frozen mode-When source or destination addresses are frozen, their value is not modified after a transfer. This mode is typically used for addressing peripheral FIFOs located at a fixed address.
- Incremented mode-When source or destination addresses are in incremented mode, after every transfer they are incremented by the number of bytes transferred.
- Decremented mode-In decremented mode, addresses are decremented by the number of bytes transferred.

The peripheral DMA registers are as follows:

- Two, 32-bit address registers (PSA and PDA) that respectively contain the source address for a read access and the destination address for a write access
- A 32-bit status register (PS) that contains information on the peripheral DMA configuration, such as the number of valid bytes in the data register, the error flag, the source and destination address mode, and so on.
- A 32-bit data register (PD) that stores data involved in a data transfer

55.4.12.2.1 Peripheral Source Address Register (PSA)

The source address register contains a pointer to a source peripheral or a memory associated with the next read data transfer. It has byte granularity.

It is based on the following:

- A 32-bit register (PSA) to store the address value
- A 2-bit register (stype) to store the source address mode (frozen, incremented, or decremented)
- A 2-bit register (ssize) to store the source target data path size (byte, half-word, or word)

Reading the register with the ldf instruction has no side effects and gives the address value of the next data that will be read by the SDMA during an ldf MD instruction. Writing the source address register may have side effects. If there is valid write data in the data register and the source address is changed, the write data is discarded. If the prefetch bit is set, a DMA read cycle is issued with the new address.

When PSA is to be written, you must specify the source target address mode, providing its size (byte, half-word, or word). This enables omission of the size field in all ldf MD instructions. When DMA performs a read cycle, its size is given by the value of the PSA source size register (ssize). If source is a memory in incremented mode, first programmed in word mode (stf PSA|SZ32|I), and if an SDMA script needs to read bytes from this memory, the size of the source target must be updated before executing new accesses. The source address mode and its size are given by labels added to the stf PSA instruction as described in the write section. The ssize and stype registers are part of the DMA status register (PS).

Writing to PSA may issue an immediate error if the source size is not compatible with the value to be written into the PSA register. For instance, writing a 2 in PSA and specifying that it is memory-accessed in word mode creates an immediate error.

55.4.12.2.2 Peripheral Destination Address Register (PDA)

The destination address register contains a pointer to a source peripheral or a memory associated with the next write data transfer. It has byte granularity.

It is based on the following:

- A 32-bit register (PDA) to store the address value
- A 2-bit register (dtype) to store the destination address mode (frozen, incremented, or decremented)
- A 2-bit register (dsize) to store the destination target data path size (byte, half-word, or word)

Reading the register with the ldf instruction has no side effects, and gives the address value of the next data that will be written by SDMA during an stfMD instruction. Writing the destination register has no side effect. Similar to the PSA register, the destination address mode and source are specified in the stf PDA instruction and may also generate an error in case of incorrect programming.

55.4.12.2.3 Peripheral Data Register (PD)

The data register of the peripheral DMA is a 32-bit register. When the destination address is correctly set up, any writing to PD will automatically flush the new input data.

The number of SDMA bytes that will be transferred is given by the PDA size register. Unlike other SDMA DMAs, PD is not a FIFO: It is not used to accumulate bytes that from the SDMA and must be packed before being sent to external memories. In read mode, and if the source address is correctly set up, an ldf instruction will empty PD. If a prefetch is required along with the instruction, the DMA will initiate a new read transfer.

Reading PD in prefetch mode only stalls the SDMA when the prefetched data is not yet available. Writing PD only stalls the SDMA if the previous write operation was not completed. As soon as the previous operation is over, the acknowledge is sent back to the SDMA RISC engine.

An error flag-part of PS-is set when an external access fails. The error is thus reported to the next SDMA instruction that involves the peripheral DMA.

55.4.12.2.4 Peripheral State Register (PS)

The state register contains the DMA state-machine value. It can be accessed in case of an error received during a transfer.

Although all PS fields can be written by an stf instruction, it is recommended to access only the error bit (to reset it). Modifying other PS fields will provide an un-guaranteed DMA behavior.

The initialization value of PS is 0, and it consists of the following structure:

Table 55-28. PS Structure

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	ssize		stype		dsize		dtype	
W																
R	0	0	0	0	0	d	e		0	0	0	0	0	n		
W																

Table 55-29. PS Field Descriptions

Field	Description
31-24	Reserved
23-22 ssize	Source Target Size. Determines the size of the read transfers on the external bus. It should match the accessed device characteristics. 00 *reserved* 01 Byte (8 bits) 10 half-word (16 bits) 11 word (32 bits)

Table continues on the next page...

Table 55-29. PS Field Descriptions (continued)

Field	Description
21-20 stype	Source address Mode. Determines whether PSA is incremented, decremented, or kept unmodified after every read from the external bus. 00 Frozen Mode 01 Incremented Mode 10 Decremented Mode 11 *reserved*
19-18 dsize	Destination Target Size. Determines the size of the write transfers on the external bus. It should match the accessed device characteristics. 00 *reserved* 01 Byte (8 bits) 10 half-word (16 bits) 11 word (32 bits)
17-16 dtype	Destination address Mode. Determines whether PDA is incremented, decremented, or kept unmodified after every write on the external bus. 00 Frozen Mode 01 Incremented Mode 10 Decremented Mode 11 *reserved*
15-11	Reserved
10 d	Direction Flag or DMA Mode. DMA is in write mode when data was written into PD by stf PD instructions, or if a previous DMA cycle on the external bus was a write access. Writing PDA or PSA does not change the DMA mode. DMA is in read mode when a previous DMA cycle was a read access, and DMA stays in read mode when data is read by the SDMA with an ldf PD instruction. Reading PDA or PSA does not change the DMA mode. 0 Read Mode 1 Write Mode
9-8 e	Error. Indicates if the previous access was acknowledged with a bus error. 00 No error was received. 01 *reserved* 10 Error mode 11 Error read
7-3	Reserved
2-0 n	number of bytes in PD

NOTE

dtype, dsize, stype, and ssize are updated when PSA and PDA are written.

55.4.12.2.5 Peripheral DMA Write (stf)-Write Mode

When written by an stf instruction, the function code bits are interpreted as follows:

Table 55-30. STF Code Bits

Register	7	6	5	4	3	2	1	0
PSA	s		p	ar	am		sz	
PDA								
PD			pdsel					
PS			pssel					

Table 55-31. STF Code Bits Field Descriptions

Field	Description
7-6 s	Functional Unit selector 11 for Peripheral DMA
5 p (PSA)	Prefetch Flag 0 no prefetch 1 automatic prefetch
4 ar (PSA/PDA)	Address Register Selector 0 PSA 1 PDA
3-2 am (PSA/PDA)	Address Mode. Determines how PSA or PDA is modified after every read or write access to the PD. 00 Frozen-Address registers are not modified after the transfer. 01 Incremented-Address registers are incremented by the number of transferred bytes. 10 Decremented-Address registers are decremented by the number of transferred bytes. 11 Updated-PSA and PDA are not modified. Either address mode is not modified, but the width of the data path is updated by the sz field.
1-0 sz	Transfer Size 00 *reserved* 01 byte (8 bits) 10 half-word (16 bits) 11 word (32 bits)
5-0 pdsel	PD access selector 001000 is the only valid option
5-0 pssel	PS access selector 111111 writes to PS 001100 only clears the error flag in PS

Due to the large number of possible stf instructions, the following table provides only a short list of all the possible write instructions:

Table 55-32. Peripheral DMA STF Instruction List

Binary	Assembly	Comments
11_00_00_01 11_00_00_10 11_00_00_11	stf Rn, PSA|SZ8 |F stf Rn, PSA|SZ16|F stf Rn, PSA|SZ32|F	• Source is a byte, half-word, or word target at the Rn address. Any further PD read instructions will trigger a byte, half-word, or word access to the source. • Source address is frozen.
11_10_00_01 11_10_00_10 11_10_00_11	stf Rn,PSA|SZ8 |F|PF stf Rn,PSA |SZ16|F|PF stf Rn,PSA |SZ32|F|PF	• Source is a byte, half-word, or word target at the Rn address. Any further PD read instructions will trigger a byte, half-word, or word access to the source. • 1, 2, or 4 bytes are *fetched* from the peripheral source. • Source address is frozen.
11_00_01_01 11_00_01_10 11_00_01_11	stf Rn, PSA|SZ8 |I stf Rn, PSA|SZ16|I stf Rn, PSA|SZ32|I	• Source is a byte, half-word, or word target at the Rn address. Any further PD read instructions will trigger a byte, half-word, or word access to the source. • Source address is in incremented mode: PSA = PSA + 1,2 or 4 after read PD.
11_10_01_01 11_10_01_10 11_10_01_11	stf Rn, PSA|SZ8 |I|PF stf Rn, PSA|SZ16|I|PF stf Rn, PSA|SZ32|I|PF	• Source is a byte, half-word, or word target at the Rn address. Any further PD read instructions will trigger a byte, half-word, or word access to the source. • Source address is in incremented mode: PSA = PSA + 1, 2, or 4 after read PD. • 1, 2, or 4 bytes are *fetched* from the peripheral source.
11_00_10_01 11_00_10_10 11_00_10_11	stf Rn, PSA|SZ8 |D stf Rn, PSA|SZ16|D stf Rn, PSA|SZ32|D	• Source is a byte, half-word, or word target at the Rn address. Any further PD read instructions will trigger a byte, half-word, or word access to the source. • Source address is in incremented mode: PSA = PSA-1,2, or 4 after read PD.
11_10_10_01 11_10_10_10 11_10_10_11	stf Rn, PSA|SZ8 |D|PF stf Rn, PSA|SZ16|D|PF stf Rn, PSA|SZ32|D|PF	• Source is a byte, half-word, or word target at the Rn address. Any further PD read instructions will trigger a byte, half-word, or word access to the source. • Source address is in incremented mode: PSA = PSA-1,2, or 4 after read PD. • 1, 2, or 4 bytes are *fetched* from the peripheral source.
11_00_11_01 11_00_11_10 11_00_11_11	stf Rn, PSA|SZ8 |U stf Rn, PSA|SZ16 |U stf Rn, PSA|SZ32 |U	• *Update* source pointer to memory, which becomes a pointer to a memory accessed in byte, half-word, or word. • PSA value is not modified by Rn. • Bytes present in PD are lost.
11_10_11_01 11_10_11_10 11_10_11_11	stf Rn, PSA|SZ8 |PF|U stf Rn, PSA|SZ16 |PF|U stf Rn, PSA|SZ32 |PF|U	• *Update* source pointer, which becomes a pointer to a target accessed in byte, half-word, or word. • PSA value is not modified by Rn. • Bytes present in PD are lost. • 1, 2, or 4 bytes are *fetched* from the memory source.
11_01_00_01 11_01_00_10 11_01_00_11	stf Rn, PDA|SZ8 |F stf Rn, PDA|SZ16|F stf Rn, PDA|SZ32|F	• Destination is a byte, half-word, or word target at the Rn address, and any further PD write instructions will trigger byte, half-word, or word access to the destination. • Destination address is frozen.
11_01_01_01 11_01_01_10 11_01_01_11	stf Rn, PDA|SZ8 |I stf Rn, PDA|SZ16|I stf Rn, PDA|SZ32|I	• Destination is a byte, half-word, or word target at the Rn address, and any further PD write instructions will trigger byte, half-word, or word access to the destination. • Destination address is in incremented mode: PDA = PDA + 1, 2, or 4 after write PD.

Table continues on the next page...

Table 55-32. Peripheral DMA STF Instruction List (continued)

Binary	Assembly	Comments
11_01_10_01 11_01_10_10 11_01_10_11	stf Rn, PDA|SZ8 |D stf Rn, PDA|SZ16|D stf Rn, PDA|SZ32|D	• Destination is a byte, half-word, or word target at the Rn address, and any further PD write instructions will trigger byte, half-word, or word access to the destination. • Destination address is in incremented mode: PDA = PDA-1, 2, or 4 after write PD.
11_01_11_01 11_01_11_10 11_01_11_11	stf Rn, PDA|SZ8 |U stf Rn, PDA|SZ16 |U stf Rn, PDA| SZ32 |U	• *Update* destination pointer to memory, which becomes a pointer to a memory accessed in byte, half-word, or word. • PDA value is not modified by Rn • bytes present in PD are lost
11_00_10_00	stf Rn, PD	• Write "dsize" bytes of Rn in PD and automatically flush to destination target
11_11_11_11	stf Rn, PS	• Write status register
11_00_11_00	stf Rn,clrefPS	• Clear error flag if set

NOTE

When writing PD, size information is not important: It is embedded in the dsize field of PDA register. If dsize is 1, 2, or 4, then one, two, or four bytes from Rn is written to the PD register, and automatically flushed out to the destination target.

55.4.12.2.6 Peripheral DMA Read (ldf)-Read Mode

When received from an ldf instruction, the function code bits are interpreted as follows.

Table 55-33. LDF Code Bits

Register	7	6	5	4	3	2	1	0
PSA	s			ar	a			
PDA								
PD			p	cpy				
PS			pssel					

Table 55-34. LDF Code Bits Descriptions

Field	Description
7-6 s	Functional Unit selector 11 for Peripheral DMA
5 p (PD)	Prefetch Flag 0 no prefetch 1 automatic prefetch
4 ar (PSA/PDA)	Address Register Selector 0 PSA

Table continues on the next page...

Table 55-34. LDF Code Bits Descriptions (continued)

Field	Description
	1 PDA
4 cpy (PD)	Copy Mode 0 standard access 1 copy mode access
3 a	Register Set selection 0 PSA or PDA 1 PD or PS
5-0 pssel	PS access selector 111111 is the only valid option to read PS

Table 55-35. Peripheral DMA LDF Instruction List

Binary	Assembly	Comments
11_0_0_0_000	ldf Rn, PSA	Reads 32-bit of PSA value
11_0_1_0_000	ldf Rn, PDA	Reads 32-bit of PDA value
11_0_0_1_000	ldf Rn, PD	Reads programmed source size bytes of PD (0-extended)
11_1_0_1_000	ldf Rn, PD|PF	Reads programmed source size bytes of PD (0-extended), and starts a prefetch at PSA address.
11_0_1_1_000	ldf Rn, PD|COPY	Starts a copy transfer from the source target at the PSA address to the destination target at the PDA address. No data transmits through Rn, but Rn contents are lost (Rn is loaded with PD temporary contents that are *not* the copied data).
11_111111	ldf Rn, PS	Reads 32-bit of PS value

NOTE

When reading PD, size information is not important: It is embedded in the ssize field of the PSA register. If ssize is 1, 2, or 4, the one, two, or four bytes is transferred from PD to Rn. Read data is 0-extended.

55.4.12.2.7 Peripheral DMA Unit Copy Mode

Like burst DMA, the peripheral DMA unit has a copy mode that is used when data transfers do not involve SDMA general registers.

Data is read from the source target at a PSA address, stored in PD, and then automatically flushed to the destination target at the PDA address. Copy mode is only available for transfers that involve two targets of the same data path width.

Since copy mode is invoked with an ldf instruction, the *loaded* general purpose register loses its previous contents. (However, the new contents are unpredictable as they depend on temporary values that are seen on the external DMA bus.)

55.4.12.2.8 Error Management

Peripheral DMA generates two kinds of errors: the immediate error that sanctioned incorrect register programming; and the error triggered by the previous access and stored in the error flag of PS until a DMA instruction is executed.

55.4.12.2.8.1 Immediate Errors

The following table lists all incorrect DMA register setups.

Table 55-36. Immediate Errors with Peripheral DMA

Rn[1:0] values	DMA instruction	Comments
0x01 0x11	stf Rn, PSA|SZ16|F stf Rn, PSA|SZ16|I stf Rn, PDA|SZ16|F stf Rn, PDA|SZ16|I	If PSA points to a half-word peripheral or to a half-word address in memory, its value must be 0 modulo 2.
0x01 0x10 0x11	stf Rn, PSA|SZ32|F stf Rn, PSA|SZ32|I stf Rn, PDA|SZ32|F stf Rn, PDA|SZ32|I	If PSA points to a word peripheral or to a word address in memory, its value must be 0 modulo 4.
PSA[1:0]-PDA[1:0]	DMA instruction	Comments
0x01 0x10 0x11	stf Rn, PSA|SZ32|U stf Rn, PDA|SZ32|U	When PDA or PSA is updated and becomes a pointer to a word address in memory, its content must be 0 modulo 4.
0x01 0x11	stf Rn, PSA|SZ16|U stf Rn, PDA|SZ16|U	When PDA or PSA is updated and becomes a pointer to a half-word address in memory, its content must be 0 modulo 2.
Read/Write PD instruction	Comments	
stf Rn,PD ldf Rn,PD	If PDA size (dsize) has never been set up before an stf PD instruction (dsize=0) If PSA size (ssize) has never been set up before an ldf PD instruction (ssize=0)	
ldf Rn,PD|CPY	Copy mode is possible only between two targets whose data path width is identical. It is P8↔P8, P16↔P16, or P32↔P32 regardless of the way the address registers are incremented.	

55.4.12.2.8.2 Data Transfer Errors

When PSA and PDA are correctly set up, the only error that may arise for an ldf PD or stf PD instruction would be the error of the previous DMA cycle.

Error handling is driven by a single consideration: When an error occurred during a data read on the DMA interface, this error should appear as a transfer error to the core when the core attempts to retrieve the data that was not successfully read from the accessed device (memory or peripheral).

When an error occurred during a write access to the DMA interface, the data is still available in PD and should not be destroyed by subsequent core accesses: The core must be warned about the error issue.

There are three error handling mechanisms for each case: Read Error (First Phase), Write Error and Read Error (Second Phase), and Copy Mode Errors handling.

55.4.12.2.8.3 Read Error (First Phase)

If an error occurred during a prefetch command, the peripheral DMA enters its ERROR READ mode (PS[9:8]=11). In this mode, the error is reported on the next ldf PD instruction and writing PSA, PDA, or PD will cancel the error flag.

The block returns no error mode and instructions are normally executed (a DMA cycle may be triggered). Similarly, initiating a copy transfer will reset the error flag and start a copy transfer. The following table details which instructions can be executed in this mode.

Table 55-37. Possibilities in ERROR READ Mode

DMA Instruction	Immediate Error	Comments
stf rn, PD stf rn, PSA (IU IPF) stf rn, PDA ldf rn,PD|COPY	NO	Error mode is reset, PSA or PDA are updated, or a write cycle is started. For the ldf PD|COPY, a copy loop is executed.
stf rn, PS	NO	PS is updated.
ldf rn, PS ldf rn, PSA ldf rn, PDA	NO	PS, PSA, and PDA could be read in ERROR READ mode without any side effects (for example, no DMA cycle is triggered).
ldf rn, PD	YES	Error of the previous read access is reported here and the peripheral DMA enters its ERROR mode.

55.4.12.2.8.4 Write Error and Read Error (Second Phase)

The peripheral DMA enters its ERROR mode (PS[9:8]=10) when the previous DMA write cycle failed, or, as explained in Read Error (First Phase), when an ldf PD is executed while the block is in ERROR READ mode. When a DMA cycle failed, address registers (PSA, PDA) are not modified and continue to point to the problematic address. In ERROR mode, stf instructions may raise an immediate error, and ldf instructions will not (as detailed in the table below).

Table 55-38. Possibilities in ERROR Mode

DMA Instruction	Immediate Error	Comments
stf rn, PD stf rn, PSA stf rn, PDA	YES	Any attempt to modify PD, PSA, or PDA will raise an immediate error, and the peripheral DMA stays in ERROR mode. When address registers are write accessed, an error is returned.
stf rn, PS	NO	This is the only way to exit the ERROR mode. PS[3] must be reset by an stf PS instruction.
ldf rn, PS ldf rn, PSA ldf rn, PDA	NO	PS, PSA, and PDA could be read in ERROR mode without any side effects (for example, no DMA cycle is triggered).
ldf rn, PD	YES	Whatever the DMA direction (read or write), an ldf rn, PD instruction will show an immediate error.

55.4.12.2.8.5 Copy Mode Errors

Because copy mode is a write access that follows a read access, there are two possible cases of bus error.

When the read access incurs a bus error, the peripheral DMA behaves exactly as described in Read Error (First Phase) and Write Error and Read Error (Second Phase) : It enters its ERROR READ mode, and so on.

When the error occurred during the write access of the copy transfer, the DMA enables the core to retrieve the data that was read because it is assumed the read from the peripheral removed the data from its source device. Therefore, the data to be flushed is still in PD. Any subsequent access to PD triggers an error to the core, which should execute its error handling procedure.

Once the ERROR mode is left (after writing to PS), it is possible for the core to retrieve the data in PD with an ldf instruction or try to flush PD contents once again (for example, when the error was due to a full FIFO and the script waited for the FIFO to be emptied) with another ldf instruction in copy mode. This latter instruction detects that there is valid data in PD, tries to flush it, and thus skips the read phase of the copy instruction. This is a different behavior from the usual stf PD instruction that overwrites PD with the selected General Purpose register contents. The same mechanism can be used any time PD holds data that is not written because of a bus error on the DMA interface; when the data was written via a copy instruction, or via the usual stf PD instruction.

55.4.12.2.8.6 Error Check Example

The following code illustrates an example checking for both immediate and data transfer errors on a store to the PD register. The first bdf instruction checks for an immediate error, but if a data transfer error occurred it is reported until the next instruction to access the Peripheral DMA. A second check of the error flags is done after the ldf PS

instruction. The value of PS here can be ignored. The act of reading any register in Peripheral DMA while it is in an error mode that returns the error to the core to set either the SF or DF flag. Any error returned on an ldf command sets the SF flag and any error returned on an stf instruction sets the DF flag. This can create a situation as shown in the example where a bus error during a DMA write which would normally be considered as a destination fault is reported as a source fault because the error was reported to the SDMA core during an ldf instruction.

Peripheral DMA Error Check

```
    clrf    0            // Clear SF and DF flags
 stf    R4, PD        // Write data to memory
bdf    error_routine    // Check for immediate error from write to PD.
ldf    r3, PS        // Read PS (PS value in R3 can be ignored)
    bsf    error_routine    // Check for bus error from "stf R4,PD"
            // SF is set because it is a ldf instruction, even though
            // the original error was a destination fault
```

55.4.12.2.9 Peripheral DMA Unit Prefetch/Flush Management

There is no flush bit because every time data is stored in PD by a stf PD instruction-assuming PDA is correctly programmed-it is automatically flushed to the destination.

An acknowledge is returned in the cycle of the DMA instruction, and the SDMA is only stalled by an instruction that addresses the peripheral DMA when the previous DMA access is not over.

55.4.12.3 OnCE and Real-Time Debug

The On-Chip Emulation block (OnCE) is the debug interface to the SDMA.

It supports the access to all core internal devices (registers, memory, and so on), and provides a set of mechanisms that control the core. The OnCE is accessed by JTAG ports at the chip's board level, or by the host via its peripheral bus.

To reduce the size of the hardware material involved, all tasks supported by the OnCE are performed on the SDMA core. The architecture of the SDMA OnCE is relatively simple and very flexible.

The commands supported by the SDMA OnCE are listed in the following sections.

55.4.12.3.1 Memory and Register Access

A set of mechanisms is provided to access SDMA memory and register locations. Both reading and writing are allowed. The access is supported if the processor is in debug mode.

Those registers can also be accessed through the ARM platform Control interface when the OnCE is controlled by the ARM platform, as described in the "Using BP" section.

55.4.12.3.2 Hardware Breakpoints

An event detection unit is implemented to support memory breakpoints. The unit watches the data exchanged between the SDMA memory bus and the core.

A debug request is sent to the core when matching conditions occur. The unit supports mixed conditions based on address range, access type, and data value. Event detection unit configuration registers are memory mapped in the SDMA space (see ARM platform Channel 0 Pointer (SDMAARM_MC0PTR)): You can modify them through a regular memory access or the ARM platform control interface.

55.4.12.3.3 Watchpoints

One output pin is provided to monitor matching trigger conditions that are defined in the event detection unit.

55.4.12.3.4 Software Breakpoints

The SDMA instruction set contains a software breakpoint. Upon executing a software breakpoint instruction, the core suspends normal execution and enters debug mode.

No hardware step execution mode is implemented in the OnCE, but this feature may be implemented at the software level with this instruction.

55.4.12.3.5 Core Control

Commands are provided to monitor and control processor activity. You can halt the core, rerun the core from another address location, and get processor status.

Any hardware breakpoint on the instruction bus is not supported, but this feature may be implemented by inserting a software breakpoints program.

55.4.13 The OnCE Controller

The OnCE controller receives commands from the ARM platform or from the JTAG controller. Each command is interpreted before being sent to the core.

55.4.13.1 OnCE Commands

A small set of commands supports the communication between the OnCE and the external world.

This command set enables you to perform any of the following tasks: control processor activity, save core context, and execute an SDMA instruction from the OnCE. Combined together, these tasks perform more complex commands.

A full OnCE command contains a 4-bit instruction (the OnCE command opcode) and a variable length data field (the OnCE data). During command execution, the OnCE data is transferred in a OnCE internal register before being exchanged with the SDMA. Some data values are also exported. This mechanism creates a link between the processor and the external world. Nine commands are defined: The following table presents their formats.

Table 55-39. OnCE Command Opcode Values

Instruction Opcode	Name	Action	Register	Data Field Size	Mode
0000	rstatus	Reads the OnCE status register	STATUS	16-bit	normal/debug
0001	dmov	Updates general register GReg1	GREG1	32-bit	debug
0010	exec_once	Runs the instruction from the SDMA instruction register	INSTRUCTION	16-bit	debug
0011	run_core	Returns to normal execution	BYPASS	1-bit	debug
0100	exec_core	Returns to normal execution via a jump instruction that specifies the new address	INSTRUCTION	16-bit	debug
0101	debug_rqst	Stops the core after execution of current instruction	BYPASS	1-bit	normal
0110	rbuffer	Reads the real time buffer	RTB	32-bit	normal/debug
0111-1110	reserved	Reserved	BYPASS	1-bit	normal/debug
1111	bypass	Bypasses TARM platform controller	BYPASS	1-bit	normal/debug

Each instruction corresponds to a specific action performed on the OnCE. The nature of the associated data field is clearly identified. The dmov command is followed by a 32-bit data value (which is a data value for the SDMA); the exec_once and the exec_core commands are followed by a 16-bit data value (which is an instruction for the SDMA); the rstatus command is followed by a 16-bit control value (which is the content of the OnCE status register); the rbuffer command is followed by a 32-bit data value. The

debug_rqst and the run_core commands are followed by a single bit data field (this is a bypass value). Finally, the bypass instruction enables the SDMA JTAG TAP controller to be daisy-chained with another JTAG TAP controller. This is a JTAG-only feature. The set of commands is simple, but enables you to perform any possible task on the SDMA during a debug process.

55.4.13.2 Sending Commands to the OnCE Controller

The JTAG access is the standard access to the OnCE, but sometimes the JTAG is not available to fix some bugs (if the chip is in production for instance), an additional access is then required. Therefore, one ARM platform access to the OnCE is provided.

55.4.13.2.1 Using the JTAG Interface

A serial access is performed through the five JTAG pins TCK, TRST, TMS, TDI, and TDO. A Test Access Port controller is provided to decode the TMS control signal.

It produces shift-enable signals (shift_ir and shift_dr), and updates enable signals (update_ir and update_dr). It is fully compliant with the IEEE 1149.1 testability (JTAG) standard.

During the shift_ir state, the command opcode is shifted into the OnCE controller (for example, the signal from the TDI pin is shifted into the command register and the TDO pin receives the signal shifted out). After transferring the four bits of the command, an update_ir signal is asserted and the command is decoded. The target data register is now clearly identified and the corresponding control signal is produced, as follows: bypass enable signal (bp_en), instruction enable signal (inst_en), data enable (data_en), and status enable signal (stat_en).

During the shift_dr state, the TDI signal is shifted into one of the following target registers: bypass register (1 bit), SDMA instruction register (16 bits), SDMA data register (32 bits), or OnCE status register (16 bits). The TDO pin is connected to the output of the selected register to receive the signals shifted out.

The JTAG access is disabled when the ARM platform access is enabled.

55.4.13.2.2 Using the ARM platform

The ARM platform access to the OnCE is not the standard access, but it is required if the JTAG is not available.

For example, if the SDMA ROM is out of use on a chip in production, and the ARM platform needs to download new code and restart the SDMA, the OnCE can easily perform this operation. This type of debug operation justifies the use of an ARM platform access to the OnCE.

To drive the OnCE, the ARM platform uses some registers contained in the ARM platform Control block of the SDMA. These registers are accessed through the ARM platform peripheral bus. Most of these registers are connected to another register in the OnCE controller. Thus, accessing one of these registers is equivalent to accessing the associated register in the OnCE controller.

The set of registers in the ARM platform Control block is listed below:

- ONCE_ENB register (1 bit, read/write)-This 1-bit register enables the ARM platform access to the OnCE. When this bit is set, the signals from the JTAG are ignored. When it is cleared, all writing operations to the following registers through the Host Control interface are ignored. This register is reset on a JTAG reset.
- ONCE_CMD register (4 bits, read/write)-This 4-bit register receives the command opcode. It is connected to the command register in the controller. A write access to this register causes the associated command to be executed on the OnCE. For example, after writing "0001" in this register, a dmov command is executed.

NOTE

On the ARM platform side, the rstatus and bypass commands are not supported. This register is reset on a JTAG reset.

- ONCE_DATA register (32 bits, read/write)-This 32-bit register is connected to the SDMA data register. This register is used when executing a dmov or rbuffer command.

NOTE

Before requesting a dmov command, the 32-bit data to transfer must be written in the ONCE_DATA register. At the end of the execution, the register is updated with GReg1 former value. This register is reset on a JTAG reset.

- ONCE_INSTR register (16 bits, read/write)-This 16-bit register is connected to the SDMA instruction register. This register is used when executing an exec_core or an exec_once command.

NOTE

Before requesting an exec_core or an exec_once command, the appropriate instruction must be written in the ONCE_INSTR register. This register is reset on a JTAG reset.

- ONCE_STAT register (16 bits, read only)-A read access to the ONCE_STAT register returns the content of the OnCE status register (OSTAT). This register is read only.
- The bypass register is not useful when the ARM platform controls the OnCE, therefore no register is defined in the ARM platform Control block to access the bypass register.

55.4.13.2.3 Conflicts Between the JTAG and the ARM platform Accesses

When ARM platform access to the SDMA OnCE is enabled (that is, when the bit in the ONCE_ENB register is set), the JTAG access is disabled. This guarantees that the block is not accessed at the same time on both sides.

It is possible to check whether the JTAG access to the SDMA OnCE is enabled from the JTAG port. When the JTAG access is disabled, the SDMA TDO always returns 1. The check requires the following steps:

- Execute a dmov command from debug mode (with neither 0xffffffff nor 0x0 as dmov value: 0x5a5a5a5a is good).
- Execute another dmov command (the value here is not important).
- The returned value from the latter dmov command should be the original one if the JTAG access is enabled; if it is 0xffffffff instead of the original input value, this means the JTAG access is disabled.

55.4.13.3 Executing a Command from the OnCE

All the commands defined in OnCE Commands can be accessed through the JTAG. The ARM platform can access all these commands except the rstatus command.

On the ARM platform side, the OnCE status is directly accessed by reading the ONCE_STAT register.

55.4.13.3.1 Nature of the Commands

Two types of commands may be distinguished. First, there are two commands that do not interact with the core: rstatus and rbuffer. Those commands may be requested at any time: They do not depend on the core status.

NOTE

Each of these commands exports a data value or a status value from the SDMA.

There are also commands that interact with the core: dmov, run_core, exec_core, exec_once, and debug_rqst. These commands are core status dependent, as follows:

- During user mode only the debug_rqst is taken into account.
- During debug mode, all these commands are taken into account except the debug_rqst. For example, an exec_once command requested while not in debug mode has no effect.

55.4.13.3.2 Execution Request

The SDMA starts executing a task in debug mode when requested by the OnCE controller. The execution starting time depends on the type of access used to communicate with the OnCE.

If the JTAG is used, the request is send after decoding the update_dr state in the TAP controller. Therefore, always cross this state when sending a command through the JTAG. If the OnCE is driven from the ARM platform side, the request is sent after detecting a write access to the ONCE_CMD register. All the registers involved in this operation must be loaded first.

The following is an example of an exec_core command execution from the ARM platform side: After writing '010' in the ONCE_CMD register, the OnCE controller asks the SDMA to execute the instruction contained in the ONCE_INSTR register. The instruction involved should be available in the ONCE_INSTR register before the beginning of the execution.

55.4.13.3.3 Command Execution

The following list shows the commands and details how each command is executed:

- rstatus command execution-The rstatus command exports the content of the OnCE status register (OSR). If the JTAG is used, the status information is captured in the OnCE status register during the capture_dr state, and shifted out after 16 TCK clock cycles in the shift_dr state. The rstatus command is not supported on the ARM platform side, but a status register is provided instead. The rstatus may be performed in both debug and user modes.
- dmov command execution-The dmov command accesses SDMA internal registers. Executing a dmov instruction exchanges the 32-bit data values between the SDMA data register and the general register GReg[1].
- If the JTAG is used, the content of GReg1 is captured in the SDMA data register during the capture_dr state, then it is shifted out after 32 TCK clock cycles in the shift_dr state. During the update_dr state, GReg1 is updated with the new, shifted-in 32-bit data value. If the OnCE is driven from the ARM platform side, the data values

contained in GReg1 and the SDMA data register are exchanged after detecting a write access to the ONCE_CMD register. The ONCE_DATA register must therefore be loaded first.

- exec_once command execution-The exec_once command executes the instruction loaded in the SDMA instruction register. The command may only be requested from debug mode. The SDMA returns to debug mode at the end of the execution.
- Change of flow instructions as well as instructions that may cause a context switch are not supported: The comprehensive list comprises done/yield/yiedge (except done 5), BF, BT, BSF, BDF, JMP, JSR, JMPR, JSRR, RET, and LOOP, as well as all the illegal instructions.

  No other command should be requested before the SDMA returns to debug mode. The SDMA status (for example, whether it is in debug mode or not) can be detected by polling with the rstatus OnCE command, monitoring the debug_mode pin, or checking the OnCE Status Register (SDMAARM_ONCE_STAT) register via the ARM platform control interface.

NOTE

Most of the instructions are single-cycle, which omits the step of polling the status. Loads and stores to DMA units are typical instructions that might require this polling.

If the JTAG is used, the 16-bit instruction is shifted in the SDMA instruction register after 16 TCK clock cycles in the shift_dr state. A request is sent to the core when the update_dr state is decoded in the TAP controller. If the OnCE is driven from the ARM platform side, the request is sent to the SDMA when detecting a write access to the ONCE_CMD register. The ONCE_INSTR register must be therefore be loaded first.

- run_core command execution-The run_core command leaves debug mode and resume normal program execution. The next instruction executed is the last instruction decoded before entering debug mode. Be sure to restore core context before re-running the core. This procedure is detailed in Restoring the Context.
- If the JTAG is used, a 1-bit bypass value is shifted in the bypass register in the shift_dr state. The SDMA is rerun when the update_dr state is decoded in the TAP controller. If the OnCE is driven from the ARM platform side, the core is rerun when detecting a write access to the ONCE_CMD register.

- exec_core command execution-The exec_core command resumes program execution from any address. The 16-bit instruction provided with the exec_core overwrites the last instruction decoded before entering debug mode. This command is designed to support change of flow instructions, so that a program execution can be restarted

from any address. After executing an exec_core command, the SDMA leaves debug mode. The exec_core command is usually used with a jmp instruction.

- If the JTAG is used, the 16-bit branch instruction is shifted in the SDMA instruction register after 16 TCK clock cycles in the shift_dr state. The SDMA is rerun when the update_dr state is decoded in the TAP controller. If the OnCE is driven from the ARM platform side, the SDMA reruns when detecting a write access to the ONCE_CMD register. The ONCE_INSTR register must therefore be loaded first. For example, to restart the SDMA from the program address 0x100, the instruction loaded should be a jump to address 0x100 instruction.
- debug_rqst command execution-The debug_rqst command puts the SDMA in debug mode. If the JTAG is used, a 1-bit bypass value is shifted in the bypass register during the shift_dr state. A debug request is sent to the SDMA when the update_dr state is decoded in the TAP controller. If the OnCE is driven from the ARM platform side, the debug request is sent when detecting a write access to the ONCE_CMD register. When the SDMA is already in debug mode, this command is simply ignored.
- rbuffer command execution-The rbuffer command exports the content of the real time buffer (RTB). If the JTAG is used, the content of the real time buffer (RTB) is captured in the SDMA data register during the capture_dr state. The register is completely shifted out after maintaining the shift_dr state during 32 TCK clock cycles. If the OnCE is driven from the ARM platform side, the content of the RTB is captured in the ONCE_DATA register after detecting a write access to the ONCE_CMD register.
- bypass command execution-This command is only available from the JTAG interface. It enables daisy-chaining of the SDMA JTAG TAP controller with other JTAG TAP controllers. This command does not change the SDMA state and can be executed in any mode (run, debug, or sleep). It selects the bypass register of the TAP controller.

55.4.13.4 Registers Descriptions

See SDMACORE, and SDMAARM, for detailed information on each register.

55.4.13.4.1 Event Cell Counter Register (ECOUNT)

The event cell counter register is a 16-bit register that contains the number of times minus one that an event detection occurs before generating a debug request.

This register should be written before attempting to use the event detection counter during an event detection process. The event cell counter register is cleared on a JTAG reset.

55.4.13.4.2 Event Cell Address Registers (EAA or EAB)

The event cell contains two address registers-the event cell address register (a), called EAA, and the event cell address register (b), called EAB. Every address register is a 16-bit register that stores a user-defined address value. This value computes one of the following address conditions: addra_cond or addrb_cond. Every address register is cleared on a JTAG reset.

55.4.13.4.3 Event Cell Address Mask Register (EAM)

The event cell address mask register is a 16-bit register that contains a user-defined address mask value. This mask is applied to the address value latched from the memory address bus before comparing addresses.

NOTE

There is a common address mask value for the two address comparators. If bit *i* of this register is set, then bit *i* of the address value latched from the memory bus does not influence the result of the address comparison. The event cell address mask register is cleared on a JTAG reset.

55.4.13.4.4 Event Cell Data Register (ED)

The event cell data register is a 32-bit register that contains a user-defined data value. This data value is an input for the data comparator, which generates the data_cond condition.

The event cell data register is cleared on a JTAG reset.

55.4.13.4.5 Event Cell Data Mask Register (EDM)

The event cell data mask register is a 32-bit register that contains a user-defined data mask value. This mask is applied to the data value latched from the memory bus before comparing data.

Setting bit *i* of the event cell data mask register means that bit *i* of the data value latched from the address bus does not influence the result of the data comparison. The event cell data mask register is cleared on a JTAG reset.

55.4.13.4.6 Real Time Buffer Register (RTB)

The real Time Buffer register is a 32-bit register that stores and retrieves run-time information without putting the SDMA in debug mode.

Refer to Real Time Buffer for more details.

55.4.13.4.7 Event Control Register (ECTL)

The event cell control register is a 16-bit register that defines cell event occurrence conditions.

The event cell control register is cleared on a JTAG reset. See also OnCE Event Detection Unit for more details.

55.4.13.4.8 Trace Buffer (TB)

The Trace Buffer register retrieves the information in the Trace Buffer.

See Trace Buffer for more details.

55.4.13.4.9 OnCE Status Register (OSTAT)

The OnCE status register is a 16-bit register that contains processor and event detection unit status. The OSTAT is a read-only register.

Refer to OnCE Status Register (SDMAARM_ONCE_STAT) for detailed description of the individual fields in the OSTAT register.

The following figure shows the OSTAT structure.

Table 55-40. OnCE Status Register (OnCE)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PST[3:0]				RCV	EDR	ODR	SWB	MST					ECDR[2:0]		

Where PST[3:0] is the SDMA core state, RCV is set when the real-time buffer (RTB) is modified. EDR, ODR, and SWB are set, respectively, when the SDMA has entered debug mode because of an external debug request, a OnCE debug_rqst command, or a software breakpoint. MST is set when the OnCE is controlled from the ARM platform control interface, and when ECDR is a three-flag set that shows the event cell condition(s) that put the core in debug mode. The OSTAT never provides more than one reason for entering debug mode.

There are two ways of accessing OSTAT content, as follows:

1. Send an rstatus command to the OnCE controller through the JTAG, or read the ONCE_STAT register through the ARM platform access. Executing the rstatus command through the JTAG can be performed in both user and debug modes.
2. Perform an SDMA read access to the location in the SDMA core memory map (OSTAT register) debug mode using the exec_once command. With this method of access, the SDMA state reflected by the PST (processor status bit) is always DATA.

The register may also be accessed by a running application.

55.4.13.5 JTAG Interface Requirements

Because the signals received from the JTAG (running on TCK) are transferred to the OnCE controller (running on the SDMA clock), a synchronization mechanism is required.

55.4.13.5.1 TCK Speed Limitation

In the JTAG top-level layer, the TDO signal is always captured on a TCK falling edge. To guarantee a stable TDO signal from the SDMA during this operation, a falling edge detection is performed on TCK.

Before being latched in the *I* flip-flop (see Figure 55-11) on TCK falling edge, the TDO signal must be stable at the input of the flip-flop. This condition is verified if the TCK period is superior to the following delay:

worst-case edge detection delay + negative-edge signal propagation delay + JTAG top-level logic propagation delay

The frequency relationship, TCK < CLK/8, limitation guarantees that all operations are performed as expected.

55.4.13.5.2 Synchronization Implementation

The figure found here shows the synchronization mechanism.

Flip-flops tck0, tck1, and tck2 perform falling- and rising-edge detections on TCK. They generate the posedge_detected and negedge_detected nets that are used to sample the TDI and TMS inputs into the respective tdi and tms flip-flops, and update the tdo flip-flop to yield the TDO output. In the design, the only signal that might go metastable is the output of the tck0 flip-flop. This signal is captured in the tck1 flip-flop and no logical operation is performed on it to minimize a metastability propagation risk.

The TDI and TMS flip-flops also cannot go metastable: The propagation time of the rising-edge detection signal through tck0, tck1, and tck2 guarantees that the TDI and TMS inputs are stable when captured in the TDI and TMS flip-flops.

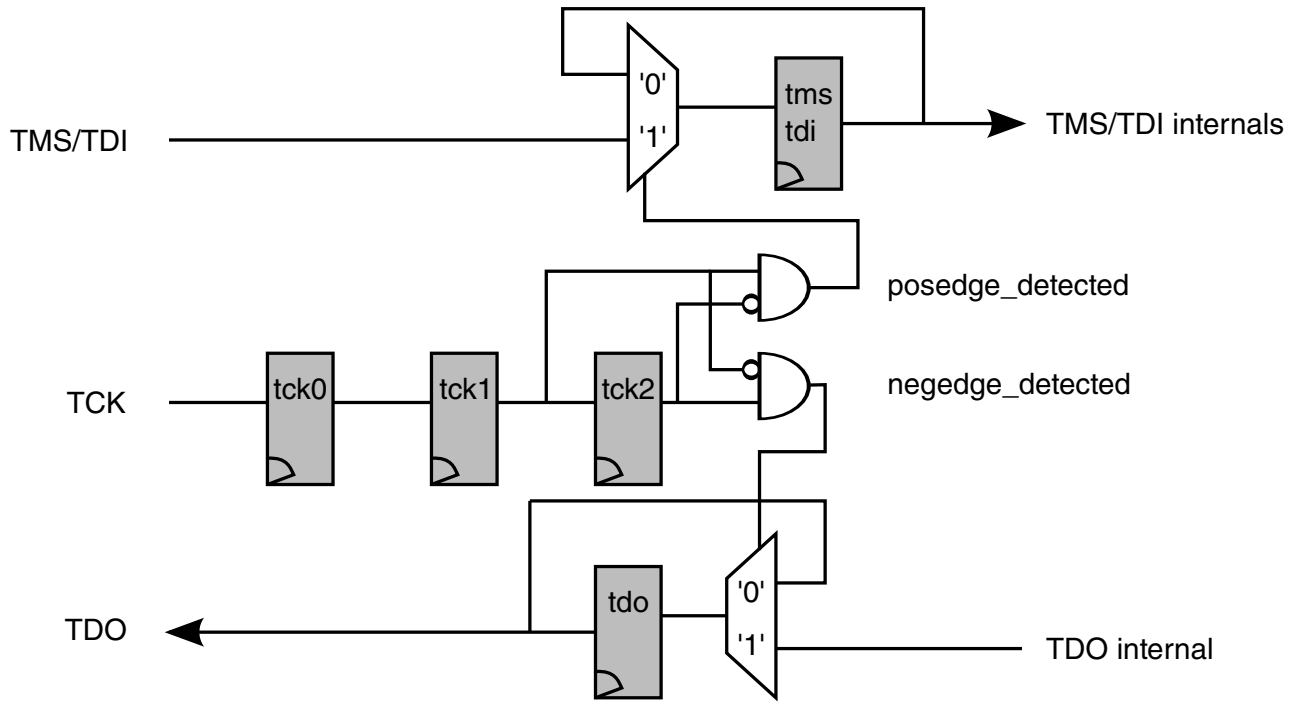


Figure 55-11. OnCE Synchronization Layer

The following figure shows synchronization timings. It takes three CLK clock cycles to synchronize TDI on the SDMA clock.

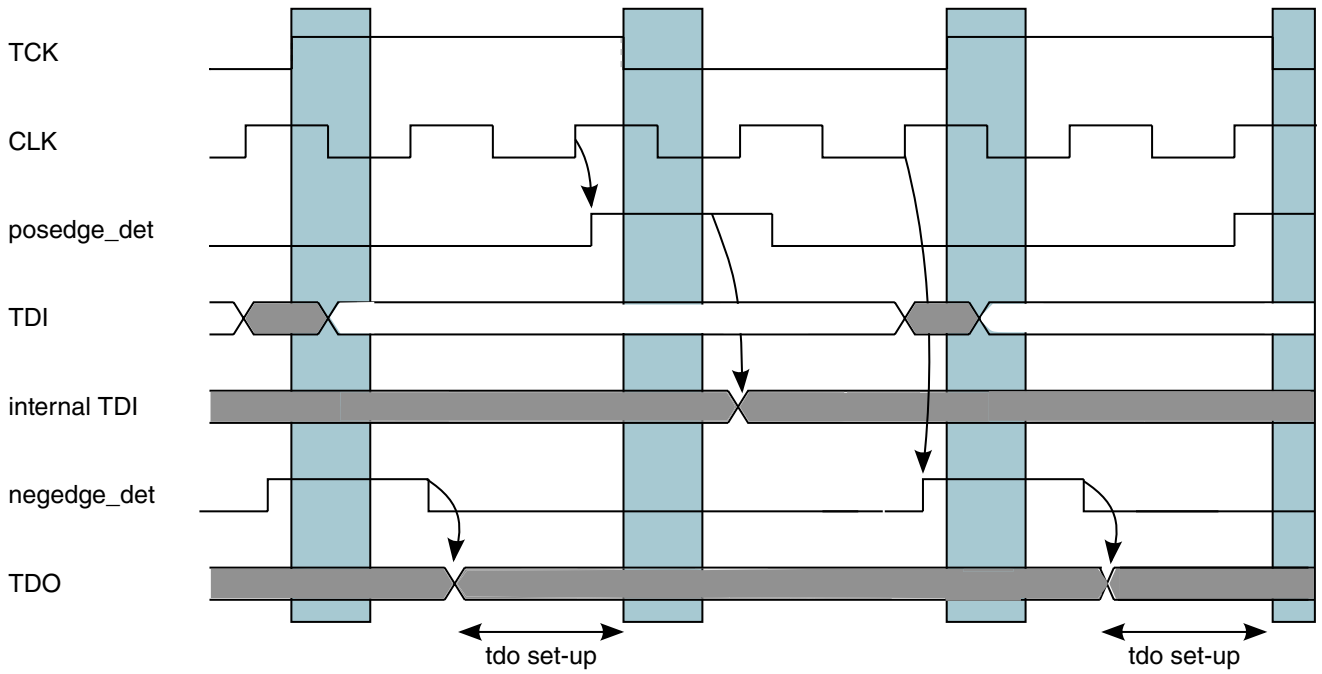


Figure 55-12. Synchronization Timings

55.4.13.5.3 JTAG Controller Start-Up Recommended Procedure

To ensure correct TAP controller initialization, it is recommended to use the following procedure:

1. Assert JTAG reset TRSTB (for example, set low).
2. Set TMS low.
3. Wait for 1 TCK clock.
4. Release JTAG reset TRSTB (for example, set high).
5. Wait for a minimum of five TCK cycles.

55.4.14 Using the OnCE

This section provides the elements necessary to run the OnCE during a debug process.

In addition to the basic set of commandsdescribed in OnCE Commands, more complex commands can be built to meet users' requirements.

55.4.14.1 Activating Clocks in Debug Mode

For power consumption issues, some clocks in the SDMA are disabled when not needed.

This is the case for instances when the SDMA is in sleep mode. Clock gating management depends on the interface used to control the OnCE.

- For the JTAG access, the SDMA clock gating must be turned off via the clk_gating_off input.
- For the ARM platform access, the SDMA clock gating is automatically turned off when the ARM platform access is enabled (see OnCE Enable (SDMAARM_ONCE_ENB)).

55.4.14.2 Getting the Current Status

Most of the commands the OnCE supports have an impact on the status of the SDMA.

It is not permissible to request the execution of an instruction on the SDMA from the OnCE while the SDMA is not in debug mode. Such a violation may cause unpredictable behavior, and it might be necessary to reset the SDMA.

Therefore, the value of the PST bits provided in the OnCE status register should always be checked before sending any request to the SDMA.

55.4.14.3 Methods of Entering Debug Mode

A debug request may be asserted at any time, but it is not always taken into account immediately. Debug mode cannot be entered in the middle of an instruction, or during the save or restore states of a context switch.

The request is ignored when the core is already in debug mode. Refer to Figure 55-4, which shows all possible transitions to the debug state, as there are several ways to enter debug mode.

55.4.14.3.1 External Debug Request During Reset

To enter debug mode after exiting reset, the external debug line has to be maintained high. This line is handled by the JTAG top-level block.

NOTE

The SDMA detects the debug requests only if the SDMA clock is running (see Activating Clocks in Debug Mode). The debug request line should be not be maintained high when the SDMA is in debug mode.

NOTE

The debug_rqst command (from the OnCE command set) is not supported during system reset.

55.4.14.3.2 Debug Request During Normal Activity

During normal activity, the SDMA enters debug mode when the following is true:

1. If the debug request line from the JTAG top-level is asserted, or
2. If the OnCE controller receives a debug_rqst command.

   The debug_rqst command can be sent by the JTAG access or by an access on the ARM platform side (if the ARM platform access is enabled).

55.4.14.3.3 Software Breakpoint Instruction

The SDMA enters debug mode at the end of the execution of a software breakpoint instruction. This instruction must be inserted in program flow executed by the core.

55.4.14.3.4 Event Detection Unit Matching Condition

If the event detection is enabled, a debug request is sent to the core after detecting a matching condition on the SDMA memory bus.

See OnCE Event Detection Unit for more details.

55.4.14.4 Executing Instructions in Debug Mode

The OnCE supports a mechanism to execute instructions in debug mode. If the SDMA is in debug mode, then the exec_once command can be used to execute an SDMA instruction from the OnCE controller. The SDMA returns to debug mode at the end of each execution.

Some instructions are not supported by the exec_once command: done/yield/yiedge (except done 5), BF, BT, BSF, BDF, JMP, JSR, JMPR, JSRR, RET, and LOOP, as well as all the illegal instructions are not supported.

NOTE

While instructions are executed in debug mode from the OnCE, the program counter of the SDMA is not incremented.

55.4.14.5 Command Sequences Examples

This section provides examples of command sequences that run the SDMA in debug mode. These sequences are available for both the ARM platform and JTAG accesses.

The following presents the syntax used in this section. The data field provided with each command is put in parenthesis with the command name. A '-' is used if the data field provided is a *don't care* value.

```
my_command(data_field);                 // executing my_command with a data field
my_command(-);                        // executing my_command with a don't care data field
```

The value returned by the command (if there is one) is referred by an assignment. In case the value returned by the command is not used, the assignment is omitted. For an ARM platform access, the value returned (it is always a data value) is obtained by reading back into the SDMA data register.

```
data_out = my_command(data_in); // returning a data value
```

To clarify the syntax, the instructions' opcodes are referred to by their names. In practice, use the corresponding 16-bit encoding.

55.4.14.5.1 Getting the SDMA Status

NOTE

Before executing any command that affects the SDMA (like dmov or exec_once), check that the SDMA is in debug mode.

Use the following snippet:

```
rstatus();        // read SDMA status until the SDMA is in debug mode
...
rstatus();
```

If the SDMA is not in debug mode, then a debug request must be generated. In this case, the SDMA enters debug mode at the end of the execution of the current instruction. Use this snippet:

```
debug_rqst(-);   // debug request
```

In the following sections, it is assumed that the SDMA was successfully put into debug mode.

55.4.14.5.2 Saving the Context

The first debug task is to save the SDMA context, which is the content of the eight general-purpose registers, the loop and PC-related registers, and the flags.

Use the general register GReg[1] as an intermediate register to export the entire context of the SDMA.

The following example shows how to save GReg[0], GReg[1], GReg[2] and GReg[3]. The sequence of commands used to export additional general registers is very similar to this.

Save GReg[0], GReg[1], GReg[2], and GReg[3]

```
GReg1_data = dmov(-);                          // the value exported is the content of
GReg[1]
exec_once("mov GReg1,GReg0");                      // puts the content of GReg[0] into
GReg[1]
GReg0_data = dmov(-);                          // the value exported is the content of
GReg[0]
exec_once("mov GReg1, GReg2");                     // puts the content of GReg[2] into
GReg[1]
GReg2_data = dmov(-);                          // the value exported is the content of
GReg[2]
exec_once("mov GReg1, GReg3");                     // puts the content of GReg[3] into
GReg[1]
GReg3_data = dmov(-);                          // the value exported is the content of
GReg[3]
```

Get the value of the internal flags (SF, DF, T, and LM), of the loop related registers (EPC and SPC), and of the PC-related registers (PC and RPC). Use a done 5, which is the formatting instruction dedicated to the debug. This instruction formats the flags and the values contained in the registers. It also writes the resulting values into the channel context memory. It should not be used when entering debug from the IDLE state (for example, with no active channel script running on the SDMA), because it will update a channel context that may belong to any channel.

```
exec_once("done 5");                   // formatting the value of flags and registers
```

At this point, the channel context should be up-to-date in memory, and debug operations should now be possible. However, the context can be exported with the following instructions:

Exporting the Context

```
dmov(ctx_base_addr);                             // loading GReg[1] with the channel
context base address
exec_once("ld GReg0,(GReg1,0)");                     // get RPC-PC into GReg0
exec_once("ld GReg1, (GReg1,1)");                    // get SPC-EPC into GReg1
Loop_data = dmov(-);                           // read back the value of Loop registers
exec_once("mov GReg1, GReg0");                     // puts the PC info into GReg1
PC_data = dmov(-);                             // reads back the content of the PC registers
```

After this sequence of operations, the entire SDMA context is exported via the OnCE.

55.4.14.5.3 Restoring the Context

At this point in the operation, restore the context of the SDMA. It can be different from the original context located in memory, and the content previously saved into the debugging application via the OnCE.

The example found hereshows how it is possible to modify the current channel context.

Modifying the Current Channel Context

```
dmov(Loop_data);                                  // put Loop former value into GReg[1]
exec_once("mov GReg0, GReg1");                        // copy to GReg[0]
dmov(PC_data);                                    // put PC former value into GReg[1]
exec_once("mov GReg2, GReg1");                        // copy to GReg[2]
dmov(ctx_base_addr);                                 // put channel context base address into
GReg[1]
exec_once("st GReg0, (GReg1,1)");                          // restore Loop context
exec_once("st GReg2, (GReg1,0)");                          // restore PC context
```

Once the context in memory is the desired context (with or without applying the previous instruction sequence), it can be restored to the *real* PC and loop registers in the SDMA core:

```
exec_once("cpShReg");                    // restore flags and PC & loop related registers
```

After this command, the SDMA core PC, RPC, SPC, EPC registers, as well as the flags contain the same data as what is stored in the context RAM for the current channel.

The following example shows how to restore the context of general registers GReg[0], GReg[1], GReg[2] and GReg[3].

Restoring the General Register Context

```
dmov(GReg3_data);                     // put GReg[3] restore value in GReg[1]
exec_once("mov GReg3, GReg1"); // restore GReg[3]
dmov(GReg2_data);                     // put GReg[2] restore value in GReg[1]
exec_once("mov GReg2, GReg1"); // restore GReg[2]
dmov(GReg0_data);                     // put GReg[0] restore value in GReg[1]
exec_once("mov GReg0, GReg1"); // restore GReg[0]
dmov(GReg1_data);                     // restore GReg[1]
```

At this point, it is possible to restart the normal program execution.

NOTE

Every SDMA core general register value can be modified by a mov instruction, which makes modification of these registers easy during debug. Unfortunately, there is no such instruction as a mov to directly modify the contents of either PCU register or flag (PC, RPC, SPC, EPC, T, LM, SF, or DF). The cpShReg instruction is meant to provide a means for changing these register contents via the context memory.

55.4.14.5.4 Accessing the Memory

In the example shown here, it is assumed that the SDMA context is entirely saved. If true, it is permissible to modify the general purpose registers during debugging activity.

To perform a memory read access, the target address is stored via the OnCE in GReg[1], then the load instruction is executed on the SDMA (the data loaded from the memory overwrites the address contained in GReg[1]), and then the result value is read back via the OnCE.

```
macro READ:            dmov(target_addr);                                  // put the target
address in GReg[1]
                    exec_once("ld GReg1,(GReg1,0)");                        // execute the
load instruction
                    res_data = dmov(-);                                 // exports the result
data value
```

For a memory write access, the target address is written in GReg[0], and the value to store is written in GReg[1]. Then the store instruction is executed on the SDMA.

```
macro WRITE:            dmov(target_addr);                                  // puts the
target address in GReg[1]
                    exec_once("mov GReg0,GReg1");                       // puts the target
address in GReg[0]
                    dmov(target_data);                                  // puts the target
data in GReg[1]
                    exec_once("st GReg1,(GReg0,0)");                        // performs the
store operation
```

This sequence is shown as an example; however, many other sequences are possible.

NOTE

This sequence of commands can also be applied to memory-mapped registers.

55.4.14.5.5 Resuming Program Execution

Before resuming program execution, it is assumed that the SDMA context is properly restored. There are two ways to restart the SDMA.

Start by executing the last instruction fetched before entering debug mode, as follows.

```
run_core(-);                          // resume execution from where we stopped before
```

If necessary, restart the execution from a different address. In this case, use the exec_core command. The data field provided with this command must be the encoding of a jump instruction.

```
exec_core("jmp start_addr"); // rerun the SDMA from another address
```

In these two examples, the SDMA exits debug mode and keeps executing the code fetched from the memory.

55.4.14.5.6 Single Stepping in RAM

To execute a program step-by-step from the RAM, insert software breakpoints in the program flow at appropriate places so that the SDMA only executes one instruction before returning to debug mode.

First, read the next instruction to execute in the RAM. Then, depending on the value of this instruction, compute the address where a software breakpoint instruction should be inserted. The instruction at the corresponding address must be saved, and, the software breakpoint instruction is inserted. After restarting the SDMA, there is only one instruction executed before meeting the software breakpoint.

The following example shows the macro functions READ and WRITE, which correspond to the sequence of commands (described above) used to access the memory.

NOTE

The data read from the memory are 32-bit values, while the instructions are 16-bit values only. This is why it is best to only use addresses divided by two when accessing the memory.

READ and WRITE Macro Functions

```
next_instr = READ(run_addr/2);                          // read the next instruction to execute
// the tool now has to compute the address where the breakpoint
// instruction should be inserted, this address is the "bkpt_addr"
instr_save = READ(bkpt_addr/2);                          // save the instruction before
overwriting
STORE("bkpt instruction",bkpt_addr/2);                        // store the bkpt instruction
in memory
exec_core("jmp run_addr");                              // rerun the SDMA
rstatus(-);                                          // wait for the SDMA to enter debug mode
...
rstatus(-);
STORE(instr_save,bpkt_addr/2);                           // restore the instruction
overwritten
```

In case of branched conditional instructions, a breakpoint instruction should be written at the two possible target addresses.

55.4.14.5.7 Single Stepping in ROM

No single-step mechanism is supported in ROM. The program code can be loaded in the RAM, where the single-step mechanism can be executed.

55.4.14.6 OnCE Event Detection Unit

The event detection unit watches signals from the data memory bus (DMBUS), which the SDMA core uses to access its RAM, ROM, and memory mapped registers.

A debug request is sent to the OnCE controller when user-defined conditions on address and/or data values are true.

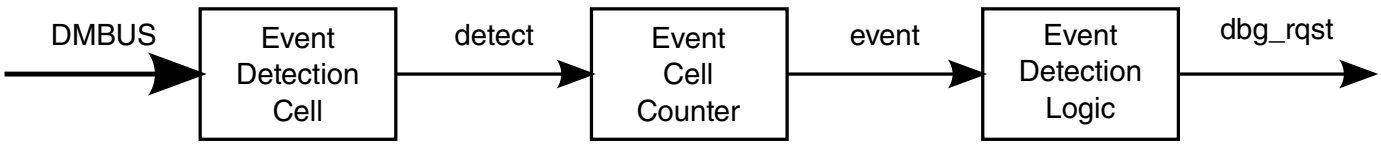


Figure 55-13. Event Detection Unit

A counter, provided with the detection cell, is decreased after an event detection. A debug request is sent to the core only when the counter reaches the value of 0. It is possible to disable the use of the counter if a debug request has to be generated after each event detection.

The event cell is the basic block that supports hardware breakpoints on an address value and/or data values coming from the SDMA memory bus. The trigger condition that generates the debug request is a mixed condition based on those values.

The following figure shows the event cell architecture. The event cell contains the address (stored in the memory address register) and the data (stored in the memory data register) used during the last memory access. There are some user-defined reference values located in memory mapped registers-the event cell addresses, the event cell address mask, the event cell data, and the event cell data mask. These registers are accessed by standard load/store instructions just like regular memory locations.

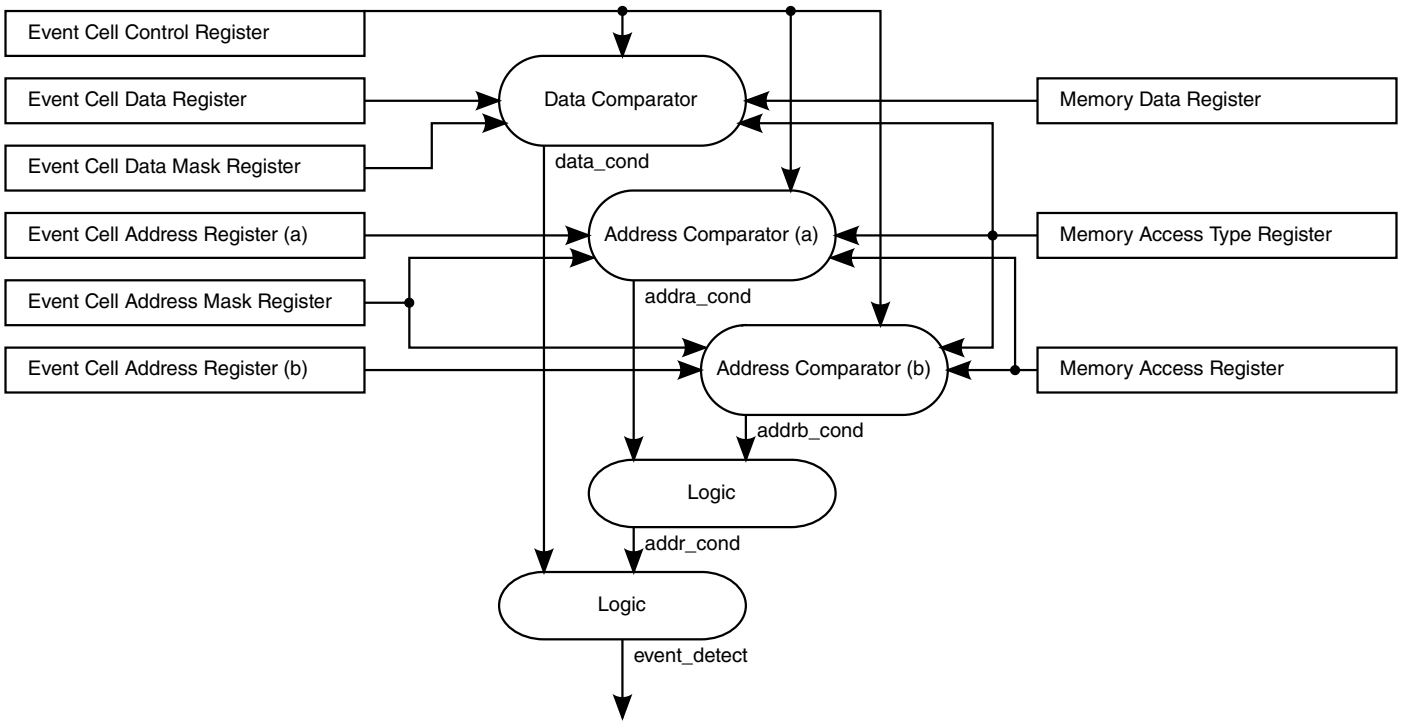


Figure 55-14. Event Cell Architecture

To define a memory breakpoint, three conditions are taken into account: The first two conditions are comparisons of the current memory address with user-defined reference addresses (these conditions are called addressA and addressB). The third condition consists of a comparison between the data received on the DMBUS and a user-defined reference data (this condition is called data). An intermediate address condition is set to express a dependency between addressA and addressB conditions.

55.4.14.7 Clock Gating and Reset

This section details how to use the clocks and handle the reset signals.

55.4.14.7.1 Clocks

Because the SDMA uses clock gating to save power, it is necessary to disable the clock gating and force the clocks to be enabled when using the OnCE.

When the OnCE is accessed through its JTAG interface, clock gating must be disabled outside the SDMA via a dedicated SDMA input port clk_gating_off. The reason why detection is not performed automatically by the SDMA internal hardware is that it would cost power to monitor activity on the JTAG interface.

When the OnCE is accessed through the ARM platform Control interface, clock gating is automatically turned off. This is done when bit 0 of the ONCE_ENB register (see OnCE Enable (SDMAARM_ONCE_ENB)) is set. A write access to this register is possible even when the OnCE clock is not running. If the ARM platform access is used, the bit in the ONCE_ENB register must be set before any attempt to access any other OnCE register.

55.4.14.7.2 Resets

The OnCE reset is different from the SDMA main reset.

Normally, activating the SDMA reset while keeping the OnCE reset inactive (when possible) enables you to reset the core without having to reprogram the OnCE.

55.4.14.8 Real Time Features

To rebuild the skeleton of a program execution, it is necessary to store the addresses of the program instructions where jumps are taken: A trace buffer is therefore provided. A real time buffer has also been added to receive data values written during a program execution.

The content of this register may be exported through JTAG ports without stopping the core.

55.4.14.8.1 Trace Buffer

The Trace Buffer is a 32-stage buffer that contains appropriate information to identify the 32 last changes of flow detected during a program execution.

The following figure shows an overview of the Trace Buffer.

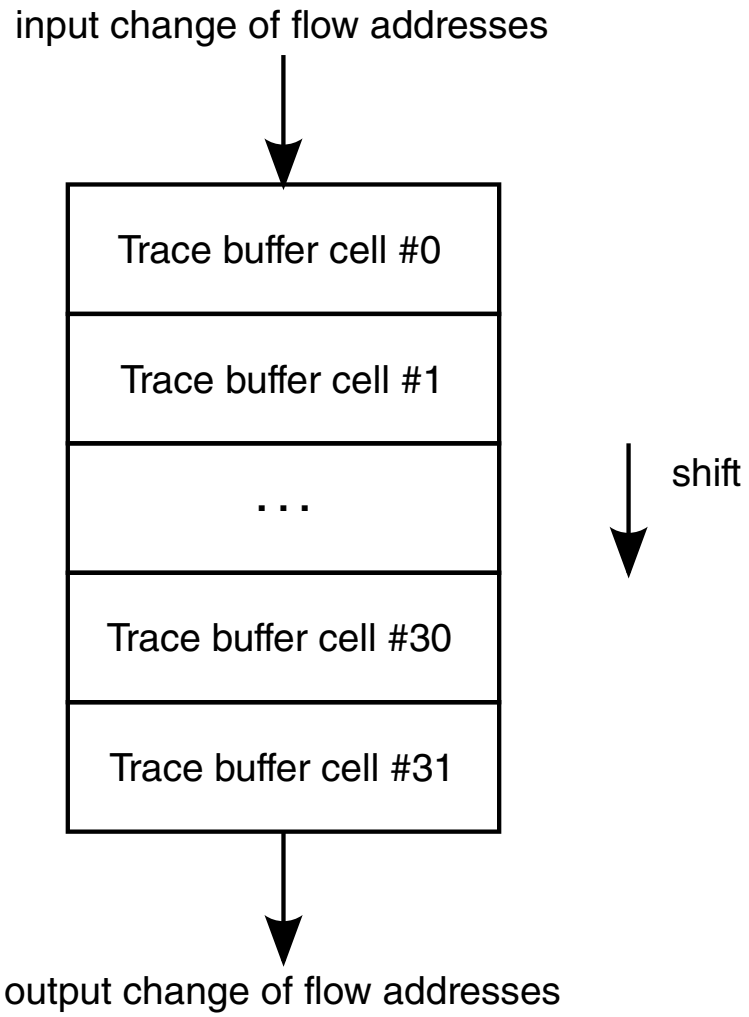


Figure 55-15. Trace Buffer

Each cell of the trace buffer contains two reference addresses and a flag. The flag is set when the addresses stored in the cell correspond to a valid change of flow; otherwise, the flag is cleared. The three most significant bits are unused.

After every change of flow detection, the address of current instruction and the address of the target instruction are stored at the top of the Trace Buffer (cell #0). The flag in the cell is set to indicate that a valid change of flow was detected. Former cell values are shifted one level down. The Trace Buffer contains the 32 last changes of flow. All the flags are reset on a software or a hardware reset, and after each transition from debug mode to user mode.

A memory mapped register of SDMA core, the Trace Buffer register (TB), is provided to read the content of the Trace Buffer. This operation should be done in debug mode. Performing a read access to the Trace Buffer register returns the content of the bottom of the Trace Buffer (cell #31). After every read access, the trace buffer is shifted one level down, and the flag at the top of the trace buffer is cleared.

A typical OnCE command sequence that retrieves the oldest change-of-flow information is a follows:

```
exec_once("mov r1, TB");                              // stores the oldest change-of-flow in
GReg1
dmov(-);                                          // retrieves GReg1 contents
```

This sequence requires the SDMA to be put in debug mode.

55.4.14.8.2 Real Time Buffer

The Real Time Buffer register (RTB) is a memory mapped register that can be accessed as a regular memory location by the SDMA core during program execution. This register is located in the OnCE.

Executing ar rbuffer command (see The OnCE Controller for further details) exports the content of this register through JTAG ports.

When a write access is performed at the memory location corresponding to the RTB, the receive flag (for example, the RCV bit) is set in the OnCE Status Register (OSR). This flag is cleared at the end of the execution of a rbuffer command.

NOTE

Every write access to the RTB memory location updates the RTB register even if the RCV flag is set. The RTB is cleared on a JTAG reset.

55.4.14.8.3 Emulation Pin

The debug_matched_event emulation pin reflects the matching condition status detected by the Event Detection Unit.

Since it can be necessary to detect conditions without triggering debug requests, it is possible to disable the generation of debug requests by the Event Detection Unit and still have the matching condition available on the emulation pin. This can be done by clearing the EN flag in the ECTL register.

55.4.14.8.4 Real-Time Debug Outputs

The table found here shows the debug signals that are available at the SDMA boundaries. Their availability at chip boundaries depends on the project.

Table 55-41. Real-Time Debug Output Pins

Pin	Description
debug_core_state[3:0]	The core_state bits reflect the state of the SDMA core. • The "Program" state is the usual instruction execution cycle. • The "Data" state is inserted when there are wait-states during a load or a store on the data bus (ld or st). • The "Change of Flow" state is the second cycle of any instruction that breaks the sequence of instructions (jumps and channel switching instructions). • The "Change of Flow in Loop" state is used when an error causes a hardware loop exit. • The "Debug" state means the SDMA is in debug mode. • The "Functional Unit" state is inserted when there are wait-states during a load or a store on the functional units bus (ldf or stf). • In "Sleep" modes, no script is running (this is the core idle state); the "after Reset" is slightly different because no context restoring phase will happen when a channel is triggered: The script located at address 0 is executed (boot operation). • The "in Sleep" states are the same as above except they do not have any corresponding channel: they are used when entering debug mode after reset; the reason is that it is necessary to return to the "Sleep after Reset" state when leaving debug mode. 0 Program 1 Data 2 Change of Flow 3 Change of Flow in Loop 4 Debug 5 Functional Unit 6 Sleep 7 Context Switch Saving Channel 8 Program in Sleep 9 Data in Sleep 10 Change of Flow in Sleep 11 Change of Flow in Loop in Sleep 12 Debug in Sleep 13 Functional Unit in Sleep 14 Sleep after Reset 15 Context Switch Restoring Channel
debug_yield	Pulse that is active when a yield (done 0) or a yieldge (done 1) instruction is executed. 0 - 1 yield/yieldge executed
debug_core_run	Active when the SDMA core is executing instructions. 0 Debug or sleep mode

Table continues on the next page...

Table 55-41. Real-Time Debug Output Pins (continued)

Pin	Description
	1 Run mode
debug_event_channel_sel	Indicates if debug_event_channel displays current channel or last received event 0- debug_event_channel[5:0] gives the number of the current channel 1- debug_event_channel[5:0] gives the number of the last received event
debug_event_channel[5:0]	Gives the number of any DMA request as soon as it is received or the number of the current channel. The value of debug_event_channel_sel indicates if debug_event_channel displays the current channel or last received event. The signal debug_event_channel_sel must be observed to determine what information is provided on debug_event_chanel at any given time.
debug_pc[13:0]	Program Counter value; it has a meaning when the core is in run mode.
debug_mode	Set when the core is in debug. 0 - 1 Core is in debug
debug_bus_error	Set when an error was received during a load or a store (ld, st, ldf, or stf instruction) and registered in SF or DF flag. 0 No error during last load/store 1 Error during last load/store
debug_bus_device[4:0]	Indicates the device or functional unit that is accessed by the current instruction. The debug_bus_device output is always valid when in sleep mode, debug mode, or executing any instruction that does not access the functional units or the memory mapped devices, "no access" is output. 0 No access 1 MSA 2 MDA 3 MD 4 MS 5 PSA 6 PDA 7 PD 8 PS 9 RESERVED 10 RESERVED 11 RESERVED 12 RESERVED 13 CA 14 CS 15 Reserved 16 Memory (RAM or ROM) 17 Memory mapped register

Table continues on the next page...

Table 55-41. Real-Time Debug Output Pins (continued)

Pin	Description
	18 Peripheral #1 19 Peripheral #2 20 Peripheral #3 21 Peripheral #4 22 Peripheral #5 23 Peripheral #6 24 Peripheral #7 25 Peripheral #8 26 Peripheral #9 27 Peripheral #10 28 Peripheral #11 29 Peripheral #12 30 Peripheral #13 31 Peripheral #14
debug_bus_rwb	Indicates the direction of the access given by debug_bus_device 0 Write access (st or stf) 1 Read access (ld or ldf)
debug_matched_dmbus	Pulse indicating the OnCE event detection unit has detected a match on the data bus during an access to memory (RAM or ROM), a memory mapped register or a peripheral that is hooked to the SDMA. 0 - 1 data bus match detected
debug_rtbuffer_write	Pulse indicating when the real-time buffer is written by the core. 0 - 1 RTB was modified
debug_evt_chn_lines[7:0]	Eight lines that generate short pulses when DMA requests are received or channels are (re)started. Every line is controlled through two parameters defined in registers Cross-Trigger Events Configuration Register 1 (SDMAARM_XTRIG_CONF1) (as described in SDMAARM). The following two parameters are available for every line: • CNF-Indicates what is monitored on the line: 0 for a channel start, 1 for a DMA request reception • NUM[5:0]-Gives the number of the DMA request or channel to monitor

The matched_event emulation pin reflects the matching condition status detected by the Event Detection Unit. Because it can be necessary to detect conditions without triggering debug requests, it is possible to disable the generation of debug requests by the Event Detection Unit and still have the matching condition available on the emulation pin. This can be done by clearing the EN flag in the ECTL register.

All real-time debug outputs are disabled by default (for example, they are stuck to 0) to avoid power consumption when they are not used. They are enabled when bit 11 (RTDOBS) of the Configuration Register (SDMAARM_CONFIG) is set. Signals provided to the system JTAG controller for SDMA debug mode status will also be enabled when the *clk_gating_off* input is asserted.

55.5 Instruction Set

55.5.1 Instruction Encoding

This section presents a short summary of the instruction codes. All context switch instructions are listed for information only; they cannot function properly out of the context switch routine.

```
x...x - don't care

rrr - destination/source general register

sss - additional source general register

bbb - general register used as address base register

ddddd - address displacement

nnnnn - bit number
uuuuuuuu - function unit command bits

pppppppp - branch displacement (signed)

iiiiiiii - 8-bit immediate

jjj - control bit to clear

ff - flag to clear
00000jjj00000000        - done (done,yield,wait)
00000jjj00000001        - notify
00000xxx00000010        - reserved
00000xxx00000011        - reserved
00000xxx00000100        - reserved
0000000000000101     - softBkpt
0000000100000101     - reserved
0000001000000101     - reserved
0000001100000101     - reserved
0000010000000101     - reserved
0000010100000101     - reserved
0000011000000101     - reserved
0000011100000101     - reserved
0000000000000110     - ret
0000000100000110     - reserved
0000001000000110     - reserved
0000001100000110     - reserved
0000010000000110     - reserved
0000010100000110     - reserved
0000011000000110     - reserved
```

```
0000011100000110     - reserved
000000ff00000111       - clrf ff
0000010000000111     - reserved
0000010100000111     - reserved
0000011000000111     - reserved
0000011100000111     - illegal
00000rrr00001000        - jmpr r
00000rrr00001001        - jsrr
00000rrr00001010        - ldrpc r
00000rrr00001011        - reserved
00000rrr000011xx          - reserved
00000rrr00010000        - revb
00000rrr00010001        - revblo
00000rrr00010010        - rorb
00000rrr00010011        - reserved
00000rrr00010100        - ror1
00000rrr00010101        - lsr1
00000rrr00010110        - asr1
00000rrr00010111        - lsl1
00000rrr001nnnnn        - bclri r,n
00000rrr010nnnnn        - bseti r,n
00000rrr011nnnnn        - btsti r,n
00000xxx10000xxx           - reserved
00000rrr10001sss           - mov
00000rrr10010sss           - xor
00000rrr10011sss           - add
00000rrr10100sss           - sub
00000rrr10101sss           - or
00000rrr10110sss           - andn
00000rrr10111sss           - and
00000rrr11000sss           - tst
00000rrr11001sss           - cmpeq
00000rrr11010sss           - cmplt
00000rrr11011sss           - cmphs
0000011011100000     - reserved
0000011011100001     - reserved
0000011011100010     - cpShReg
0000011011100011     - reserved
0000011011100100     - reserved
0000011011100101     - reserved
0000011011100110     - reserved
0000011011100111     - reserved
00000xxx11101xxx           - reserved
00000xxx11110xxx           - reserved
00000xxx11111xxx           - reserved
00001rrriiiiiiii                - ldi r,i
00010rrriiiiiiii                - xori r,i
00011rrriiiiiiii                - addi r,i
00100rrriiiiiiii                - subi r,i
00101rrriiiiiiii                - ori r,i
00110rrriiiiiiii                - andni r,i
00111rrriiiiiiii                - andi r,i
01000rrriiiiiiii                - tsti r,i
01001rrriiiiiiii                - cmpeqi r,i
01010rrrdddddbbb          - ld r,(d,b)
01011rrrdddddbbb          - st r,u
01100rrruuuuuuuu          - ldf r,u
01101rrruuuuuuuu          - stf r,u
011100xxxxxxxxxx                - reserved
011101xxxxxxxxxx                - reserved
011110ffnnnnnnnn        - Loop ff flags are reset
01111100pppppppp     - bf pc=pc+signed(pppppppp)+1
01111101pppppppp     - bt pc=pc+signed(pppppppp)+1
01111110pppppppp     - bsf pc=pc+signed(pppppppp)+1
01111111pppppppp     - bdf pc=pc+signed(pppppppp)+1
10aaaaaaaaaaaaaa     - jmp absolute
11aaaaaaaaaaaaaa     - jsr absolute
```

55.5.2 SDMA Instruction Set

This section describes all the useful instructions from the SDMA set.

Table 55-42. SDMA Instruction List

Instruction	Description	Page
ADD	Addition	ADD (Addition)
ADDI	Add with Immediate Value	ADDI (Add with Immediate Value)
AND	Logical AND	AND (Logical AND)
ANDI	Logical AND with Immediate Value	ANDI (Logical AND with Immediate Value)
ANDN	Logical AND NOT	ANDN (Logical AND NOT)
ANDNI	Logical AND with Negated Immediate Value	ANDNI (Logical AND with Negated Immediate Value)
ASR1	Arithmetic Shift Right by 1 Bit	ASR1 (Arithmetic Shift Right by 1 Bit)
BCLRI	Bit Clear Immediate	BCLRI1 (Bit Clear Immediate)
BDF	Conditional Branch if Destination Fault	BDF (Conditional Branch if Destination Fault)
BF	Conditional Branch if False	Functional Units Programming Model
BSETI	Bit Set Immediate	BSETI (Bit Set Immediate)
BSF	Conditional Branch if Source Fault	BSF (Conditional Branch if Source Fault)
BT	Conditional Branch if True	BT (Conditional Branch if True)
BTSTI	Bit Test immediate	BTSTI (Bit Test immediate)
CLRF	Clear ARM platform flags	CLRF (Clear ARM platform flags)
CMPEQ	Compare for Equal	CMPEQ (Compare for Equal)
CMPEQI	Compare with Immediate for Equal	CMPEQI (Compare with Immediate for Equal)
CMPHS	Compare for Higher or Same	CMPHS (Compare for Higher or Same)
CMPLT	Compare for Less Than	CMPLT (Compare for Less Than)
cpShReg	Update Context of PCU Registers and Flags	cpShReg (Update Context of PCU Registers and Flag)
DONE	DONE, Yield	DONE (DONE, Yield)
ILLEGAL	ILLEGAL Instruction	ILLEGAL (ILLEGAL Instruction)
JMP	Unconditional Jump Immediate	JMP (Unconditional Jump Immediate)
JMPR	Unconditional Jump	JMPR (Unconditional Jump)
JSR	Unconditional Jump to Subroutine Immediate	JSR (Unconditional Jump to Subroutine Immediate)
JSRR	Unconditional Jump to Subroutine	JSRR (Unconditional Jump to Subroutine)
LD	Load Register	LD (Load Register)
LDF	Load Register from Functional Unit	LDF (Load Register from Functional Unit)

Table continues on the next page...

Table 55-42. SDMA Instruction List (continued)

Instruction	Description	Page
LDI	Load Register with Immediate Value	LDI (Load Register with Immediate Value)
LDRPC	Load from RPC to Register	LDRPC (Load from RPC to Register)
LOOP	Hardware Loop	LOOP (Hardware Loop)
LSL1	Logical Shift Left by 1 Bit	LSL1 (Logical Shift Left by 1 Bit)
LSR1	Logical Shift Right by 1 Bit	LSR1 (Logical Shift Right by 1 Bit)
MOV	Logical Move	MOV (Logical Move)
NOTIFY	Notify to ARM platform	NOTIFY (Notify to ARM platform)
OR	Logical OR	OR (Logical OR)
ORI	Logical OR with Immediate Value	ORI (Logical OR with Immediate Value)
RET	Return from Subroutine	RET (Return from Subroutine)
REVB	Reverse Byte Order	REVB (Reverse Byte Order)
REVBLO	Reverse Low Order Bytes	Reverse Low Order Bytes(REVBLO)
ROR1	Rotate Right by 1 Bit	ROR1 (Rotate Right by 1 Bit)
RORB	Rotate Right by 1 Byte	RORB (Rotate Right by 1 Byte)
SOFTBKPT	Software Breakpoint	SOFTBKPT (Software Breakpoint)
ST	Store Register	ST (Store Register)
STF	Store Register in Functional Unit	STF (Store Register in Functional Unit)
SUB	Subtract	SUB (Subtract)
SUBI	Subtract with Immediate	SUBI (Subtract with Immediate)
TST	Test with Zero	TST (Test with Zero)
TSTI	Test Immediate	TSTI (Test Immediate)
XOR	Logical Exclusive OR	XOR (Logical Exclusive OR)
XORI	Exclusive OR with Immediate	XORI (Exclusive OR with Immediate)

55.5.2.1 ADD (Addition)

Operation:

```
GReg[r] ← GReg[s] + GReg[r]

T ← (GReg[r] == 0)
```

Assembler:

```
Syntax: add r,s

Example: add 0,3
```

ADD GReg[3] and GReg[0] and store the result in GReg[0]

CPU Flags: T

Cycles: 1

Description: Performs the ADDition of the source general register *s* and the destination general register *r*, and stores the result in the destination general register *r*. The T flag is set if the result of the operation is 0. It is cleared if the result is not 0.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	0	0	1	1	s	s	s

Instruction Fields:

rrr / sss - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

55.5.2.2 ADDI (Add with Immediate Value)

Operation:

```
GReg[r] ← GReg[r] + immediate

T ← (GReg[r] == 0)
```

Assembler:

```
Syntax: addi r,immediate

Example: add 6,112
```

ADD GReg[6] and decimal value 112 and store the result in GReg[6]

CPU Flags: T

Cycles: 1

Description: Adds a 0-extended immediate value to a general register; stores the result in the general register. The flag T is set when the result of the operation is 0; otherwise, it is cleared. The immediate value is the low-order byte of the instruction and has a maximum value of 255 (0xFF).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiiiiii - immediate value:

```
00000000 - 0

00000001 - 1

 ...

11111110 - 254

11111111 - 255
```

55.5.2.3 AND (Logical AND)

Operation:

```
GReg[r] ← GReg[s] & GReg[r]
```

Assembler:

```
Syntax: and r,s

Example: and 1,2
```

AND GReg[1] and GReg[2] and store the result in GReg[1]

CPU Flags: Unaffected

Cycles: 1

Description: Performs the AND of the source general register *s* and the destination general register *r*, and stores the result in the destination general register *r*.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	0	1	1	1	s	s	s

Instruction Fields:

rrr / sss - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

55.5.2.4 ANDI (Logical AND with Immediate Value)

Operation:

```
GReg[r] ← GReg[r] & immediate
```

Assembler:

```
Syntax: andi r,immediate

Example: andi 7,45
```

AND GReg[7] and decimal value 45 and store the result in GReg[7]

CPU Flags: unaffected

Cycles: 1

Description: Performs an AND between a 0-extended immediate value and a general register; stores the result in the general register. The immediate value is the low-order byte of the instruction and has a maximum value of 255 (0xFF).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	1	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiiiiii - immediate value:

```
00000000 - 0

00000001 - 1

...

11111110 - 254

11111111 - 255
```

55.5.2.5 ANDN (Logical AND NOT)

Operation:

```
GReg[r] ← ~GReg[s] & GReg[r]
```

Assembler:

```
Syntax:andn r,s

Example: andn 3,4
```

AND GReg[3] and NOT GReg[4] (bit inverted) and store the result in GReg[3]

CPU Flags: Unaffected

Cycles: 1

Description: Performs the AND of the negation of the source general register *s* and the destination general register *r*, and stores the result in the destination general register *r*.

Instruction Format:

Table 55-43. Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	0	1	1	0	s	s	s

Instruction Fields:

rrr /sss - destination register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

55.5.2.6 ANDNI (Logical AND with Negated Immediate Value)

Operation:

```
GReg[r] ← GReg[r] & ~immediate
```

Assembler:

```
Syntax: andni r,immediate

Example: andni 0,2
```

AND GReg[0] and decimal value -3 (inverted 32-bit value 2) and store the result in GReg[0]

CPU Flags: unaffected

Cycles: 1

Description: Performs an AND between the negation of a 0-extended 8-bit immediate value and a general register; stores the result in the general register. The immediate value is the low-order byte of the instruction and has a maximum value of 255 (0xFF).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiiiiii - immediate value:

```
00000000 - 0

00000001 - 1

...

11111110 - 254

11111111 - 255
```

55.5.2.7 ASR1 (Arithmetic Shift Right by 1 Bit)

Operation:

```
GReg[r]:{b31,b30,...,b1,b0} ← GReg[r]:{b31,b31,b30,...,b1}
```

Assembler:

```
Syntax: asr1 r

Example: asr1 3
```

divide by 2 the signed value of GReg[3] and store the result in GReg[3]

CPU Flags: Unaffected

Cycles: 1

Description: Shift the bits of any general register to the right and keep the same sign: The left bit (bit 31) is kept untouched.

Instruction Format:

Table 55-44. Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	1	0	1	1	0

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

55.5.2.8 BCLRI1 (Bit Clear Immediate)

Operation:

```
GReg[r]:{b31,...,b(i+1),0,b(i-1),...,b0} ← GReg[r]:{b31,...,b(i+1),b(i),b(i-1),...,b0}
```

Assembler:

```
Syntax: bclri r,i

Example: bclri 1,12
```

clear bit 12 in GReg[1]

CPU Flags: Unaffected

Cycles: 1

Description: Clear the bit of register r specified by the 5-bit immediate field

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Table continues on the next page...

0	0	0	0	0	r	r	r	0	0	1	i	i	i	i	i

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiii - immediate value:

```
00000 - 0

00001 - 1

...

11110 - 30

11111 - 31
```

55.5.2.9 BDF (Conditional Branch if Destination Fault)

Operation:

```
if (DF == 1) PC ← PC + 1 + displacement else PC ← PC + 1
```

Assembler:

```
Syntax:bdf label

Example: bdf LLL
```

Jump to LLL if DF is set, or go to the next instruction if DF is cleared; the displacement value is calculated by the assembler.

CPU Flags: Unaffected

Cycles: 2 when the branch is done, 1 otherwise

Description: If flag DF is set, jump to the new address that is calculated by adding the sign-extended 8-bit displacement to the next PC address. If flag DF is cleared, no jump is performed: The next instruction is located at the next PC address.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	p	p	p	p	p	p	p	p

Instruction Fields:

ppppppp - signed displacement field:

```
00000000 - 0

00000001 - 1

...

01111110 - 126

01111111 - 127

10000000 - (-128)

10000001 - (-127)

...

11111110 - (-2)

11111111 - (-1)
```

55.5.2.10 BF (Conditional Branch if False)

Operation:

```
if (T == 0)

PC ← PC + 1 + displacement

else

PC ← PC + 1
```

Assembler:

```
Syntax: bf label

Example: bf LLL
```

Jump to LLL if T is cleared, or go to the next instruction if T is set. The displacement value is calculated by the assembler.

CPU Flags: Unaffected

Cycles: 2 when the branch is done, 1 otherwise

Description: Conditional branch: If flag T is cleared, jump to the new address that is calculated by adding the sign-extended 8-bit displacement to the next PC address. If flag T is set, no jump is performed: The next instruction is located at the next PC address.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	0	0	p	p	p	p	p	p	p	p

Instruction Fields:

ppppppp - signed displacement field:

```
00000000 - 0

00000001 - 1

...

01111110 - 126

01111111 - 127

10000000 - (-128)

10000001 - (-127)

...

11111110 - (-2)

11111111 - (-1)
```

55.5.2.11 BSETI (Bit Set Immediate)

Operation:

```
GReg[r]:{b31,...,b(i+1),1,b(i-1),...,b0} ← GReg[r]:{b31,...,b(i+1),b(i),b(i-1),...,b0}
```

Assembler:

```
Syntax: bseti r,i

Example: bseti 6,5
```

Set bit 5 in GReg[6]

CPU Flags: Unaffected

Cycles: 1

Description: Sets bit number i in the selected General Register.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	1	0	i	i	i	i	i

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiii - bit number field:

```
00000 - 0

00001 - 1

...

11110 - 30

11111 - 31
```

55.5.2.12 BSF (Conditional Branch if Source Fault)

Operation:

```
if (SF == 1) PC ← PC + 1 + displacement else PC ← PC + 1
```

Assembler:

```
Syntax: bsf label

Example: bsf LLL
```

Jump to LLL if SF is set, or go to the next instruction if SF is cleared. The displacement value is calculated by the assembler.

CPU Flags: Unaffected

Cycles: 2 when the branch is done, 1 otherwise

Description: Conditional branch: If flag SF is set, jump to the new address that is calculated by adding the sign-extended 8-bit displacement to the next PC address. If flag SF is cleared, no jump is performed: The next instruction is located at the next PC address.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	0	p	p	p	p	p	p	p	p

Instruction Fields:

pppppppp - signed displacement field:

```
00000000 - 0

00000001 - 1

...

01111110 - 126

01111111 - 127

10000000 - (-128)

10000001 - (-127)

...

11111110 - (-2)

11111111 - (-1)
```

55.5.2.13 BT (Conditional Branch if True)

Operation

```
if (T == 1)

PC ← PC + 1 + displacement

else

PC ← PC + 1
```

Assembler

```
Syntax: bt label

bt LLL
```

Jump to LLL if T is set, or go to the next instruction if T is cleared. The displacement value is calculated by the assembler.

CPU Flags: Unaffected

Cycles: 2 when the branch is done, 1 otherwise

Description: Conditional branch: If flag T is set, jump to the new address that is calculated by adding the sign-extended 8-bit displacement to the next PC address. If flag T is cleared, no jump is performed: The next instruction is located at the next PC address.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	0	1	p	p	p	p	p	p	p	p

pppppppp - signed displacement field:

```
00000000 - 0

00000001 - 1

...

01111110- 126

01111111 - 127

10000000 - (-128)

10000001 - (-127)

...

11111110 -  (-2)

11111111 - (-1)
```

55.5.2.14 BTSTI (Bit Test immediate)

Operation:

```
T ← GReg[r]:b(i)
```

Assembler:

```
Syntax: btsti r,i

Example: btsti 2,29
```

Test bit 29 in GReg[2] and copy its value in flag T

CPU flags: T

Cycles: 1

Description: T is loaded with the value of bit number i from the selected general register.

Instruction Format:

Table 55-45. Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	1	1	i	i	i	i	i

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiii - bit number field:

```
0000 - 0

0001 - 1

...

11110 - 30

11111 - 31
```

55.5.2.15 CLRF (Clear ARM platform flags)

Operation:

```
 if (ff%2 == 0)

SF ← 0

if (ff/2 == 0)

DF ← 0
```

Assembler:

```
Syntax: clrf ff

Example: clrf 2
```

Clear flag SF and keep flag DF unchanged

CPU Flags: SF, DF

Cycles: 1

Description: Clears a selection of the ARM platform fault flags: SF, DF, both SF and DF or none can be cleared.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	f	f	0	0	0	0	0	1	1	1

Instruction Fields:

ff - flags field:

```
00 - clear SF and clear DF

01 - clear DF

10 - clear

SF 11 - no clear
```

55.5.2.16 CMPEQ (Compare for Equal)

Operation:

```
T ← (GReg[s] == GReg[r])
```

Assembler:

```
Syntax: cmpeq r,s

Example: cmpeq 7,5
```

Compare GReg[7] and GReg[5] and set flag T if they are equal

CPU flags: T

Cycles: 1

Description: Subtracts the destination general register *r* from the source general register *s*, and sets T if the result is 0, clears T if the result is not 0.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	1	0	0	1	s	s	s

Instruction Fields:

rrr / sss - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

55.5.2.17 CMPEQI (Compare with Immediate for Equal)

Operation:

```
T ← (GReg[r] == immediate)
```

Assembler:

```
Syntax: cmpeqi r,immediate

Example: cmpeqi 2,13
```

Compare GReg[2] and decimal value 13 and set flag T if they are equal

CPU Flags: T

Cycles: 1

Description: Subtracts the 0-extended 8-bit immediate value from the general register, and sets T if the result is 0, clears T if the result is not 0. The immediate value is the low-order byte of the instruction.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - destination register field:

```
000 - GReg[0]
```

```
001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiiiiii - immediate value:

```
00000000 - 0

00000001 - 1

...

11111110 - 254

11111111 - 255
```

55.5.2.18 CMPHS (Compare for Higher or Same)

Operation:

```
T ← (GReg[r] ≥ GReg[s])
```

Assembler:

```
Syntax: cmphs r,s

Example: cmphs 0,1
```

Compare GReg[0] and GReg[1] and set flag T if GReg[0] is higher than or equal to GReg[1]

CPU Flags: T

Cycles: 1

Description: Compares the destination general register *r* and the source general register *s*, and sets T if the destination general register *r* is higher than or equal to the source general register *s*, clears T otherwise. The comparison is unsigned.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	1	0	1	1	s	s	s

Instruction Fields:

rrr / sss - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

55.5.2.19 CMPLT (Compare for Less Than)

Operation:

```
T ← (GReg[r] < GReg[s])
```

Assembler:

```
Syntax: cmplt r,s

Example: cmplt 7,4
```

Compare GReg[7] and GReg[4] and set flag T if GReg[7] is lower than GReg[4]

CPU Flags: T

Cycles: 1

Description: Compares the destination general register *r* and the source general register *s*, and sets T if the destination general register *r* is lower than the source general register *s*, clears T otherwise. The comparison is signed.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	1	0	1	0	s	s	s

rrr / sss - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]
```

```
011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

55.5.2.20 cpShReg (Update Context of PCU Registers and Flag)

Assembler:

```
Syntax: cpShReg
```

CPU Flags: none

Cycles: 1

Description: SF, RPC, T, PC,LM, EPC, DF, and SPC registers are updated according to the value of their corresponding bits in the context memory. This instruction must only be used in debug mode via the OnCE. It reverses the done *5* operation.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	1	0	1	1	1	0	0	0	1	0

55.5.2.21 DONE (DONE, Yield)

Operation:

```
if (jjj&6 == 2) HE[CCR] ← 0

if (jjj == 3) HI[CCR] ← 1

if (jjj == 4) EP[CCR] ← 0


if ((jjj == 0) && (NCP > CCP)) CCR ← NCR

else if ((jjj == 1) && (NCP >= CCP))

CCR ← NCR

else

CCR ← NCR
```

(CCR stands for Current Channel Register; NCR stands for Next Channel Register)

Assembler:

```
Syntax: done jjj

Example: done 3
```

Clear HE bit for the current channel, send an interrupt to the ARM platform for the current channel and reschedule.

CPU Flags: Unaffected

Cycles: Variable if a context switch is done, 1 otherwise

Description: Clears one of the channel enabling bits (HE or EP for the corresponding channel number) if required. Sends an interrupt to the corresponding ARM platform by setting the appropriate flag, if required (HI for the corresponding channel number). Reschedules according to the mode and the NCP (Next Channel Priority) and CCP (Current Channel Priority) values. According to the scheduling decision, the NCR (Next Channel Register) is copied to the CCR (Current Channel Register) and channel contexts are switched. If several channels with the same highest priority are pending, they are ordered by their number from 31 down to 0. The higher number is selected (for example, channel 26 is selected if channels 3, 12, 14, and 26 with the same highest priority are pending). If no flag is modified, the reschedule can allow the replacement of the current channel by another channel with a priority strictly greater than the current channel priority (yield). Or, it can allow the replacement of the current channel by another channel with a priority greater than or equal to the current channel priority (yieldge). In the latter case, the selected channel will always be the first one with the same priority, starting from channel number 31 down to channel 0 (the current channel does not belong to the set of selectable channels).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	j	j	j	0	0	0	0	0	0	0	0

jjj - Channel Flags field:

000 - No channel flags affected: Reschedule only if the next channel priority is greater than current channel priority (yield)

001 - No channel flags affected: Reschedule only if the next channel priority is greater than or equal to the current channel priority (yieldge)

010 - Clear HE for the current channel and reschedule 011 - Clear HE, set HI for the current channel and reschedule 100 - Clear EP for the current channel and reschedule

101 - Reserved for debug to copy relevant registers into context memory

110 - RESERVED

111 - RESERVED

For the scheduling rules, refer to Scheduler Functional Description. Every possible done instruction is further described as follows:

- done 0/yield is executed by a channel script when it accepts preemption by a higher priority channel;
- done 1/yieldge is executed by a channel script when it accepts preemption by a higher priority channel and it also accepts a roll-up with other channels that have the same priority;
- done 2 is executed by a channel script that was triggered by a ARM platform start via the Channel Start (SDMAARM_HSTART) register, when its task is completed and it requires termination;
- done 3 is executed by a channel script that was triggered by a ARM platform start via the Channel Start (SDMAARM_HSTART) register, when its task is completed, it requires termination and it needs to trigger an interrupt to the ARM platform upon closure;
- done 4 is executed by a channel script that was triggered by a DMA request, when its task is completed and it requires termination;
- done 5 is used in debug mode only; it copies the PCU registers and flags to the context memory of the current channel;

55.5.2.22 ILLEGAL (ILLEGAL Instruction)

Operation:

```
PC ← 0001
```

Assembler:

```
Syntax: illegal
```

CPU Flags: Unaffected

Cycles: 2

Description: Jumps to the Illegal instruction routine located at address 0001. All unauthorized instructions result in an Illegal instruction behavior; however, the ILLEGAL instruction must be used to guarantee software compatibility with future versions of the SDMA.

Instruction Format

Table 55-46. Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1

55.5.2.23 JMP (Unconditional Jump Immediate)

Operation:

```
PC ← absolute_address
```

Assembler:

```
Syntax: jmp label

Example: jmp LLL
```

The assembler translates the label to the exact address

CPU Flags:Unaffected

Cycles: 2

Description: Jumps to the absolute address contained the lower 14 bits of the instruction (the PC is a 14-bit register).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	a	a	a	a	a	a	a	a	a	a	a	a	a	a

aaaaaaaaaaaaaa - address field:

```
00000000000000 - 0

00000000000001 - 1

...

11111111111110 - 16382

11111111111111 - 16383
```

55.5.2.24 JMPR (Unconditional Jump)

Operation:

```
PC ← GReg[r]
```

Assembler:

```
Syntax: jmpr r

Example: jmpr 0
```

Jump to address stored in GReg[0]

CPU Flags: Unaffected

Cycles: 2

Description: Jumps to the absolute address contained in a General Register.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	0	1	0	0	0

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

55.5.2.25 JSR (Unconditional Jump to Subroutine Immediate)

Operation:

```
RPC ← PC + 1

PC ← absolute_address
```

Assembler:

```
Syntax: jsr r

Example:jsr LLL
```

Jumps to subroutine starting at LLL; the assembler translates the label to exact address

CPU Flags: Unaffected

Cycles: 2

Description: Jumps to the subroutine located at the absolute address contained the lower 14 bits of the instruction (the PC is a 14-bit register).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	a	a	a	a	a	a	a	a	a	a	a	a	a	a

aaaaaaaaaaaaaa - address field:

```
00000000000000 - 0

00000000000001 - 1

...

11111111111110 - 16382

11111111111111 - 16383
```

55.5.2.26 JSRR (Unconditional Jump to Subroutine)

Operation:

```
RPC ← PC + 1

PC ← GReg[r]
```

Assembler:

```
Syntax: jsrr r

Example:jsrr 5
```

Jumps to subroutine located at address stored in GReg[5]

CPU Flags: Unaffected

Cycles: 2

Description: Jumps to the subroutine at address contained in a General Register

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	0	1	0	0	1

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

55.5.2.27 LD (Load Register)

Operation:

```
GReg[r] ← [GReg[b] + displacement]

if (transfer_error)

SF ← 1

else

SF ← 0
```

Assembler:

```
Syntax: ld r,(b,displacement)

Example: ld 1,(2,23)
```

Loads data into GReg[1]; the data is located at address obtained by adding decimal value 23 to GReg[2]

CPU Flags: SF

Cycles: 2+n where n is 0 for ROM, RAM or memory mapped registers, and n is the number of wait-states of the peripheral for a peripheral access

Description: Adds a 5-bit 0-extended displacement to a base address in General Register b; the result is the address of the data to fetch on the DM bus. The data received from the bus is stored in the destination General Register r. If an error occurs during the transfer, the flag SF is set, else it is cleared.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	r	r	r	d	d	d	d	d	b	b	b

rrr / bbb - register field:

```
000 - GReg[0]

001 - GReg[1]

...

111 - GReg[7]
```

ddddd - displacement value:

```
00000 - 0

00001 - 1

...

11111 - 31
```

55.5.2.28 LDF (Load Register from Functional Unit)

Operation:

```
GReg[r] ← [fu_address]

if (transfer_error)

SF ← 1

else

SF ←0
```

fu_address is an 8-bit field and depends on addressed functional unit

Assembler:

```
Syntax: ldf r,fu_address

Example: ldf 0,13
```

Loads data coming from the Burst DMA register MD into GReg[0]; it is a 32-bit access with no prefetch

CPU Flags: SF

Cycles: 1+n where n is the number of wait-states that may be inserted by the functional unit

Description: Sends an 8-bit address on the Functional Unit Bus (FU bus) and stores the data received from the bus in the destination General Register r. If an error occurs during the transfer, the flag SF is set, else it is cleared.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	r	r	r	f	f	f	f	f	f	f	f

See the following sections for more details of the LDF instruction usage with each functional unit:

- Burst DMA Read (ldf) for Burst DMA
- Peripheral DMA Read (ldf)-Read Mode for Peripheral DMA

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

ffffffff - functional unit source register and action (unspecified values are reserved):

```
00000000 - MSA

00000100 - MDA

00001001 - MD byte

00001010 - MD halfword

00001011 - MD word

00001100 - MS

00101001 - MD byte - prefetch

00101010 - MD halfword - prefetch

00101011 - MD word - prefetch

01000000 - DSA
```

```
11000000 - PSA

11001000 - PD

11010000 - PDA

11011000 - PD in copy mode (rrr contents are lost)

11101000 - PD - prefetch next data

11111111 - PS
```

55.5.2.29 LDI (Load Register with Immediate Value)

Operation:

```
GReg[r] ← immediate
```

Assembler:

```
Syntax: ldi r,immediate

Example: ldi 6,1
```

loads decimal value 1 into GReg[6]

CPU Flags: Unaffected

Cycles: 1

Description: Stores a 0-extended immediate value in a General Register. The immediate value is the low-order byte of the instruction and has a maximum value of 255 (0xFF).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]
```

110 - GReg[6]

111 - GReg[7]

iiiiiiii - immediate value:

00000000 - 0

00000001 - 1

...

11111110 - 254

11111111 - 255

55.5.2.30 LDRPC (Load from RPC to Register)

Operation:

GReg[r] ← RPC

Assembler:

Syntax: ldrpc r

Example: ldrpc 3

copies RPC to GReg[3]

CPU Flags: Unaffected

Cycles: 1

Description: Stores the contents of the RPC in a General Register. That instruction may be used to have more than one level of subroutines.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	0	1	0	1	0

Instruction Fields:

rrr - register field:

000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

```
101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

55.5.2.31 LOOP (Hardware Loop)

Operation:

```
if (ff%2 == 0)
   SF ← 0
if (ff/2 == 0)
  DF ← 0
if ((GReg[0] == 0) || (SF == 1) || (DF == 1))
   PC ← PC + loop_size + 1
else
   {
      SPC ← PC + 1
      EPC ← PC + loop_size + 1
      LM ← 1
      PC ← PC + 1
   }
```

during every instruction execution in the loop:

```
if ((SF == 1) || (DF == 1))
 {
    LM ← 0
    PC ← EPC
  }
else if ((PC + 1) == EPC)
 {
    GReg[0] ← GReg[0] - 1
    if (GReg[0] == 0)
      {
         LM ← 0
         PC ← EPC
```

```
       }
     else
         PC ← SPC
   }
else
  PC ← nextPC(instruction)
```

after the execution of the last instruction of the loop body:

```
if (GReg[0] == 0)
   T ← 1
else
   T ← 0
```

Assembler:

```
Syntax: loop n{,ff}
Example: loop 3,1
```

Executes GReg[0] times the instructions comprised between PC+1 and PC+3 (included); ff=1 clears the DF flag before starting the loop. When omitted, the ff field is set to 0 (clearing both SF and DF).

CPU Flags: LM[1:0], T

Cycles: 2 when the loop count (GReg[0]) is 0 or SF or DF is set at loop start, 1+1 when the loop starts but exits abnormally (SF or DF set inside the loop which adds 1 cycle to the offending load or store to jump to EPC), 1 when the loop is executed normally

Description: The loop instruction executes a sequence of instructions several times. The number of times is given by the contents of GReg[0], the loop counter. SDMA will jump to the first instruction after the end of the loop if the value in GReg[0] is 0. Otherwise the SDMA enters loop mode. It sets the most significant bit of the LM flag that will only be reset once the last instruction of the last loop is executed. The instructions in the loop are executed GReg[0] times. The management of fault flags (SF and DF) is as follows. When entering the hardware loop, SF and DF can be cleared according to the ff field of the instruction. After that operation, if any flag is still set the loop will not be executed. The SDMA will jump to the first instruction after the end of the loop without entering loop mode. During the execution of the loop, if any fault flag is set by a LD, LDF, ST, or STF instruction, the SDMA will immediately exit loop mode and jump to the first instruction after the end of the loop. In that case, GReg0 is not decremented for that last piece of the loop body execution (even if the SF or DF flag is set at the last instruction of the loop body). The T flag reflects the state of GReg[0] after the end of the loop, which is an indicator of the complete execution of the loop. If the loop exited because of an error (SF

or DF set), GReg[0] will not be 0 at the end of the loop, hence T will be cleared. If the loop executes without fault, GReg[0] will be 0 at the end of the loop, hence T will be set. The boundary case when a source or destination fault occurs at the last instruction of the last loop is considered as an anticipated exit of the loop, which causes the T flag to be cleared. If the last instruction executed before leaving the hardware loop also tries to modify the T flag, the flag is updated according to the value of GReg[0], NOT according to the result of the last executed instruction.

Limitations:

1. 1. Jump instructions (JMP, JMPR, JSR, JSRR, BF, BT, BSF, BDF) are not allowed inside the hardware loop.
2. 2. GReg[0] cannot be written to inside the hardware loop (it can be read).
3. 3. The empty loop (0 instruction in the body) is forbidden.
4. 4. If GReg[0] == 0 at the start of the loop, which causes a jump to EPC, the T flag is not updated.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	0	f	f	n	n	n	n	n	n	n	n

Instruction Fields:

```
ff - flags field:

00 - clear SF and clear DF

01 - clear DF

10 - clear SF

11 - no clear
```

nnnnnnnn - loop size

```
00000000 - empty loop: forbidden value

00000001 - 1 instruction in the loop

00000010 - 2 instructions in the loop

...

11111111 - 255 instructions in the loop
```

55.5.2.32 LSL1 (Logical Shift Left by 1 Bit)

Operation:

```
GReg[r]:{b30,...,b1,b0,0} ← GReg[r]:{b31,b30,...,b1,b0}
```

Assembler:

```
Syntax: lsl1 r

Example: lsl1 2
```

multiplies by 2 the value in GReg[2]

CPU Flags: Unaffected

Cycles: 1

Description: Shift the bits of any General Register to the left. The right bit (bit 0) is set to 0. No overflow is detected by the hardware.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	1	0	1	1	1

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

55.5.2.33 LSR1 (Logical Shift Right by 1 Bit)

Operation:

```
GReg[r]:{0,b31,b30,...,b1} ← GReg[r]:{b31,b30,...,b1,b0}
```

Assembler:

```
Syntax: lsr1 r

Example: lsr1 4
```

divides by 2 the unsigned value contained in GReg[4]

CPU Flags: Unaffected

Cycles: 1

Description: Shift the bits of any General Register to the right. The left bit (bit 31) is set to 0.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	1	0	1	0	1

Instruction Fields:

rrr - destination register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

55.5.2.34 MOV (Logical Move)

Operation:

```
GReg[r] ← GReg[s]
```

Assembler:

```
Syntax: mov r,s

Example: mov 4,0
```

copies GReg[0] to GReg[4]

CPU Flags: Unaffected

Cycles: 1

Description: Move the contents of the source General Register s to the destination General Register r.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	0	0	0	1	s	s	s

Instruction Fields:

rrr / sss - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

55.5.2.35 NOTIFY (Notify to ARM platform)

Operation:

```
 if (jjj & 4 == 0)
 {
   if (jjj&2 == 2)
      HE[CCR] ← 0
   if (jjj&1== 1)
       HI[CCR] ← 1
  }
else if (jjj == 4)
   EP[CCR] ← 0
else
```

(CCR stands for Current Channel Register)

Assembler:

```
Syntax: notify jjj

Example: notify 3
```

clears the HE bit for the current channel and sends an interrupt to the Host for the current channel

CPU Flags: Unaffected

Cycles: 1

Description: Clears one of the channel enabling bits (HE or EP for the corresponding channel number) if required, sends an interrupt to the corresponding ARM platform by setting the appropriate flag if required (HI for the corresponding channel number).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	j	j	j	0	0	0	0	0	0	0	1

jjj - Channel Flags field:

```
000 - unused

001 - set HI for the current channel

010 - clear HE for the current channel

011 - clear HE, set HI for the current channel

100 - clear EP for the current channel

101 - RESERVED

110 - RESERVED

111 - RESERVED
```

55.5.2.36 OR (Logical OR)

Operation:

```
GReg[r] ← GReg[s] | GReg[r]
```

Assembler:

```
Syntax: or r,s

Example: or 3,6
```

ORs GReg[3] and GReg[6] and stores the result in GReg[3]

CPU Flags: Unaffected

Cycles: 1

Description: Performs the OR of the source General Register s and the destination General Register r, and stores the result in the destination General Register r.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	0	1	0	1	s	s	s

Instruction Fields:

rrr / sss - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

55.5.2.37 ORI (Logical OR with Immediate Value)

Operation:

```
GReg[r] ← GReg[r] | immediate
```

Assembler:

```
Syntax: ori r,immediate

Example: ori 1,56
```

ORs GReg[1] and the decimal value 56 and stores the result in GReg[1]

CPU Flags: unaffected

Cycles: 1

Description: Performs an OR between a 0-extended 8-bit immediate value and a General Register; stores the result in the General Register. The immediate value is the low-order byte of the instruction and has a maximum value of 255 (0xFF).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiiiiii - immediate value:

```
00000000 - 0

00000001 - 1

...

11111110 - 254

11111111 - 255
```

55.5.2.38 RET (Return from Subroutine)

Operation:

```
PC ← RPC
```

Assembler:

```
Syntax: ret
```

CPU Flags: Unaffected

Cycles: 2

Description: Return from subroutine.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

55.5.2.39 REVB (Reverse Byte Order)

Operation:

```
GReg[r]:{B3,B2,B1,B0} ← GReg[r]:{B0,B1,B2,B3}
```

Assembler:

```
Syntax: revb r

Example:  revb 5
```

reverses bytes order in GReg[5]

CPU Flags: Unaffected

Cycles: 1

Description: Reverse the byte order of any General Register.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	1	0	0	0	0

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

55.5.2.40 Reverse Low Order Bytes(REVBLO)

Operation:

```
GReg[r]:{B3,B2,B0,B1} ← GReg[r]:{B3,B2,B1,B0}
```

Assembler:

```
Syntax: revblo r

Example: revblo 0
```

reverses low order bytes in GReg[0]

CPU Flags: Unaffected

Cycles: 1

Description: Reverse both low order bytes of any General Register.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	1	0	0	0	1

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

55.5.2.41 ROR1 (Rotate Right by 1 Bit)

Operation:

```
GReg[r]:{b0,b31,b30,...,b1} ← GReg[r]:{b31,b30,...,b1,b0}
```

Assembler:

```
Syntax: ror1 r

Example: ror1 3
```

rotates bits to the right in GReg[3]

CPU Flags: Unaffected

Cycles: 1

Description: Rotate the bits of any General Register to the right.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	1	0	1	0	0

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

55.5.2.42 RORB (Rotate Right by 1 Byte)

Operation:

```
GReg[r]:{B0,B3,B2,B1} ← GReg[r]:{B3,B2,B1,B0}
```

Assembler:

```
Syntax: rorb r

Example: rorb 2
```

rotates bytes to the right in GReg[2]

CPU Flags: Unaffected

Cycles: 1

Description: Rotate the bytes of any General Register to the right.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	1	0	0	1	0

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

55.5.2.43 SOFTBKPT (Software Breakpoint)

Operation:

Stops the current script and enters debug mode

Assembler:

```
softbkpt
```

CPU Flags: Unaffected

Description: When the core executes this instruction, it has the same effect as receiving a debug request from the OnCE or via the external debug request input: the script execution halts, the PCU enters its debug state and waits for the OnCE commands that are described in OnCE and Real-Time Debug.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

55.5.2.44 ST (Store Register)

Operation:

```
[GReg[b] + displacement] ← GReg[r]

if (transfer_error)
```

```
 DF ← 1

 else

DF ← 0
```

Assembler:

```
Syntax: st r,(b,displacement)

Example: st 7,(0,9)
```

stores the value from GReg[7] into memory at address obtained by adding decimal value 9 to GReg[0]

CPU Flags: DF

Cycles: 2+n where n is 0 for ROM, RAM or memory mapped registers, and n is the number of wait-states of the peripheral for a peripheral access

Description: Adds a 5-bit 0-extended displacement to a base address in General Register b; the result is the address of the data to store on the DM bus. The data sent on the bus comes from the source General Register r. If an error occurs during the transfer, the flag DF is set, else it is cleared.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	1	r	r	r	d	d	d	d	d	b	b	b

Instruction Fields:

rrr / bbb - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

ddddd - displacement value:

```
00000 - 0

00001 - 1
```

```
...

11111 - 31
```

55.5.2.45 STF (Store Register in Functional Unit)

Operation:

```
[fu_address] ← GReg[r] 0

if (transfer_error) 0

DF ← 1 0

else 0

DF ← 0
```

fu_address is an 8-bit field

Assembler:

```
Syntax: stf r,fu_address

Example: stf 3,0x2B
```

stores the 32-bit contents of GReg[3] to the Burst DMA register MD; waits until the flush to external memory is completed

CPU Flags: DF

Cycles: 1+n where n is the number of wait-states that may be inserted by the functional unit

Description: Sends an 8-bit address on the Functional Unit Bus (FU bus) and sends the contents of the source General Register r on the bus. If an error occurs during the transfer, the flag DF is set, else it is cleared.

Table 55-47. Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	r	r	r	f	f	f	f	f	f	f	f

See the following sections for more details of the STF instruction usage with each functional unit:

- Burst DMA Write (stf) for Burst DMA
- Peripheral DMA Write (stf)-Write Mode for Peripheral DMA

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

ffffffff - functional unit destination register and action (unspecified values are reserved):

00000000 - MSA in incremented mode

00000100 - MDA in incremented mode

00001001 - MD byte

00001010 - MD halfword

00001011 - MD word

00001100 - clear MS error flag

00001111 - MS

00010000 - MSA in frozen mode

00010100 - MDA in frozen mode

00011000 - MD in copy mode - number of words in rrr

00100000 - MSA in incremented mode - start prefetch

00101000 - MD no data - flush

00101001 - MD byte - flush

00101010 - MD halfword - flush

00101011 - MD word - flush

00110000 - MSA in frozen mode - start prefetch

11000001 - PSA in frozen mode - 8-bit data width

11000010 - PSA in frozen mode - 16-bit data width

11000011 - PSA in frozen mode - 32-bit data width

11000101 - PSA in incremented mode - 8-bit data width

11000110 - PSA in incremented mode - 16-bit data width

11000111 - PSA in incremented mode - 32-bit data width

11001000 - PD

11001001 - PSA in decremented mode - 8-bit data width

11001010 - PSA in decremented mode - 16-bit data width

11001011 - PSA in decremented mode - 32-bit data width

11001100 - clear PS error flag

11001101 - PSA data width becomes 8-bit

11001110 - PSA data width becomes 16-bit

11001111 - PSA data width becomes 32-bit

11010001 - PDA in frozen mode - 8-bit data width

11010010 - PDA in frozen mode - 16-bit data width

11010011 - PDA in frozen mode - 32-bit data width

11010101 - PDA in incremented mode - 8-bit data width

11010110 - PDA in incremented mode - 16-bit data width

11010111 - PDA in incremented mode - 32-bit data width

11011001 - PDA in decremented mode - 8-bit data width

11011010 - PDA in decremented mode - 16-bit data width

11011011 - PDA in decremented mode - 32-bit data width

11011101 - PDA data width becomes 8-bit

11011110 - PDA data width becomes 16-bit

11011111 - PDA data width becomes 32-bit

11100001 - PSA in frozen mode - 8-bit data width - prefetch data

11100010 - PSA in frozen mode - 16-bit data width - prefetch data

11100011 - PSA in frozen mode - 32-bit data width - prefetch data

11100101 - PSA in incremented mode - 8-bit data width - prefetch data

11100110 - PSA in incremented mode - 16-bit data width - prefetch data

11100111 - PSA in incremented mode - 32-bit data width - prefetch data

11101001 - PSA in decremented mode - 8-bit data width - prefetch data

11101010 - PSA in decremented mode - 16-bit data width - prefetch data

11101011 - PSA in decremented mode - 32-bit data width - prefetch data

11101101 - PSA data width becomes 8-bit - prefetch data

11101110 - PSA data width becomes 16-bit - prefetch data

11101111 - PSA data width becomes 32-bit - prefetch data

11111111- PS

55.5.2.46 SUB (Subtract)

Operation:

```
GReg[r] ← GReg[r] - GReg[s]

T ← (GReg[r] == 0)
```

Assembler:

```
Syntax: sub r,s

Example: sub 4,7
```

SUBtracts GReg[7] from GReg[4] and stores the result in GReg[4]

CPU Flags: T

Cycles: 1

Description: Subtracts the source General Register s from the destination General Register r, and stores the result in the destination General Register r. The T flag is set if the result of the operation is 0; it is cleared if the result is not 0.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	0	1	0	0	s	s	s

Instruction Fields:

rrr / sss - register fields:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

55.5.2.47 SUBI (Subtract with Immediate)

Operation:

```
GReg[r] ← GReg[r] - immediate

T ← (GReg[r] == 0)
```

Assembler:

```
Syntax: sub r,immediate

Example: sub 1,255
```

SUBtracts decimal value 255 from GReg[1] and stores the result in GReg[1]

CPU Flags: T

Cycles: 1

Description: Subtracts a 0-extended 8-bit immediate value from a General Register; stores the result in the General Register. The flag T is set when the result of the operation is 0; otherwise, it is cleared. The immediate value is the low-order byte of the instruction and has a maximum value of 255 (0xFF).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiiiiii - immediate value:

```
00000000 - 0

00000001 - 1

...

11111110 - 254

11111111 - 255
```

55.5.2.48 TST (Test with Zero)

Operation:

```
T ← ((GReg[s] & GReg[r]) != 0)
```

Assembler:

```
Syntax: tst r,s

Example: tst 2,3
```

ANDs GReg[2] and GReg[3] and sets T if the result is non-null

CPU Flags: T

Cycles: 1

Description: Performs the AND of the source General Register s and the destination General Register r, and sets T if the result is not 0, clears T if the result is 0.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	1	0	0	0	s	s	s

Instruction Fields:

rrr / sss - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

55.5.2.49 TSTI (Test Immediate)

Operation:

```
T ← ((GReg[r] & immediate) != 0)
```

Assembler:

```
Syntax: tsti r,immediate

Example: tsti 5,13
```

ANDs GReg[5] and decimal value 13 and sets T if the result is non-null

CPU Flags: T

Cycles: 1

Description: Performs the AND of a 0-extended 8-bit immediate value and the destination General Register r, and sets T if the result is not 0, clears T if the result is 0. The immediate value is the low-order byte of the instruction and has a maximum value of 255 (0xFF).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - destination register field:

```
000 - GReg[0]
```

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]

iiiiiiii - immediate value:

00000000 - 0

00000001 - 1

...

11111110 - 254

11111111 - 255

55.5.2.50 XOR (Logical Exclusive OR)

Operation:

GReg[r] ← GReg[s] ^ GReg[r]

Assembler:

Syntax: xor r,s

Example: xor 0,3

XORs GReg[0] and GReg[3] and stores the result in GReg[0]

CPU Flags: Unaffected

Cycles: 1

Description: Performs the eXclusive OR of the source General Register s and the destination General Register r, and stores the result in the destination General Register r.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	0	0	1	0	s	s	s

Instruction Fields:

rrr / sss - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

55.5.2.51 XORI (Exclusive OR with Immediate)

Operation:

```
GReg[r] ← GReg[r] ^ immediate
```

Assembler:

```
Syntax: xori r,immediate

Example: xor 7,5
```

XORs GReg[5] and decimal value 5 and stores the result in GReg[7]

CPU Flags: Unaffected

Cycles: 1

Description: Performs an eXclusive OR between a 0-extended 8-bit immediate value and a General Register; stores the result in the General Register. The immediate value is the low-order byte of the instruction and has a maximum value of 255 (0xFF).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]
```

```
100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiiiiii - immediate value:

```
00000000 - 0

00000001 - 1

...

11111110 - 254

11111111 - 255
```

55.5.2.52 YIELD, YIELDGE (DONE, Yield)

By default, unsupported assembler syntax. Can be aliased to the corresponding done instructions (yield = done 0; yieldge = done 1). Refer to the done instruction description DONE (DONE, Yield) .

55.6 Software Restrictions

55.6.1 Unsupported Burst DMA Access Sequence

The SDMA does not support triggering a pre-fetch followed by a flush of the Burst DMA without reading or writing any data. If the flush occurs while the background pre-fetch DMA operation is still in progress, it could result in un-defined behavior.

An example of the sequence which could result in undefined results is shown in the following example:

Instruction sequence not supported

```
stf r1, MSA|PF          ; Update source address, triggers data pre-fetch in the
                        ; background
mov R0,R0               ; Execute multiple assembly instructions, none of which
                        ; read
mov R0,R0               ; or write data to/from MD
stf MD|SZ0|FL           ; Flush FIFO without writing data. If the pre-fetch is still
                        ; in progress when this instruction is executed, there
                        ; could be undefined operation
```

A work-around to avoid any undesirable results is to first read MD to ensure the pre-fetch is complete before the flush is attempted.

Work-Around to previous example

```
stf r1, MSA|PF          ; Update source address, triggers data pre-fetch.
mov R0,R0               ; Execute multiple assembly instructions, none of which
                        ; read
mov R0,R0               ; or write data to/from MD
ldf r2, MD              ; dummy read of MD to ensure pre-fetch is complete
                        ; before the next instruction
stf MD|SZ0|FL           ; Flush FIFO without writing data
```

55.7 Application Notes

55.7.1 Data Structures for Boot Code and Channel Scripts

SDMA boot code downloads the different channel contexts and the scripts that will be executed on SDMA channels during the application.

The boot code is run after reset when channel 0 is started by the ARM platform. The boot code is also known as channel 0 script.

The boot code is based on the Channel Control Block (CCB) and Buffer Descriptor (BD) mechanisms that are data structures located into the ARM platform memory space. With these data structures, it is possible to instruct SDMA to download scripts and contexts but also to dump a context or a script to a destination data buffer. Channel scripts also use the CCB and BD data structures to pass instructions and/or pointers to data to be copied.

The format, processing, and field definition of the CCB and BD are defined and performed entirely by the software script rather than the SDMA hardware. An overview of the format and structure is provided here, but for complete details refer to the SDMA software documentation (see SDMA Scripts).

The CCB and BD data structures are accessed by SDMA using DMA and processed by the SDMA scripts. The ROM contains common sub-routines for processing these data structures which may be called by the bootload and channel scripts.

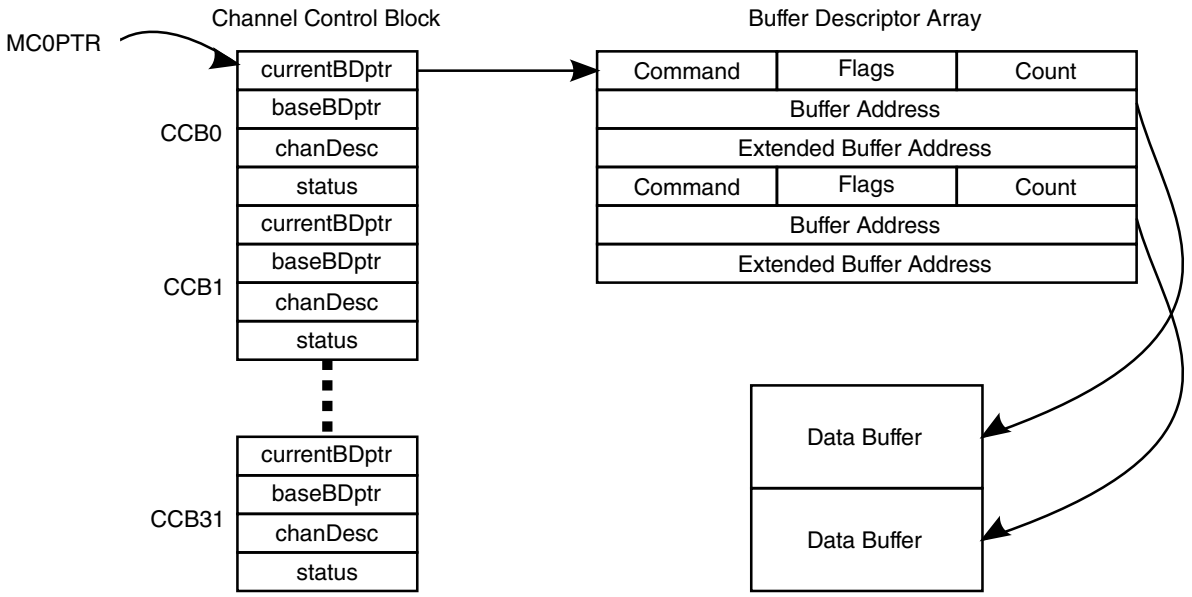


Figure 55-16. Data Structures Layout

The previous figure shows an example how these data structures are linked to pass command and pointers to data buffers. The SDMA's MC0PTR register holds the base address of the Channel 0 Control Block (CCB0). The Channel 0 control block holds a pointer to the array of buffer descriptors. The buffer descriptors are used to tell the channel 0 (boot channel) what to do as described Buffer Descriptor Format.

55.7.1.1 Buffer Descriptor Format

Buffer descriptors are three longs (32-bit words) in size as, shown in the figure found here.

A buffer descriptor describes the properties of the data buffer it points to. The buffer descriptors can be used for linear or circular data buffers in the ARM platform processor memory. The CCB contains a pointer to the base BD as well as the current BD.

Table 55-48. Buffer Descriptor

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Command								-	-	L	R	I	C	W	D	Count															
Buffer Address																															
Extended Buffer Address																															

Table 55-49. Buffer Descriptor Field Descriptions

Field	Description
31-24 Command	Command. The command field is used to differentiate operations performed within a script when the script accesses this particular buffer descriptor. The use of this field can be defined by the script. The command values defined for the bootload script are defined in Buffer Descriptor Commands for Bootload scripts. Refer to the individual script definition in script library documents in SDMA Scripts for command field definitions for other scripts.
23	Reserved
22	Reserved
21 L	Last Buffer Descriptor: This bit is set in SDMA IPC scripts to indicate to the receiving Core that the transfer has ended. Whenever the source finishes transferring the count it wanted to transfer, it sets LAST_BIT in the destination BD, to let the destination know that transfer is over.This bit also tells the destination software that when it processes the destination BDs, they need not process any BD after the BD with the LAST_BIT set.For example, when the DSP prepares a single buffer descriptor with count equals to 25 and ARM platform prepares a single buffer descriptor with count equals 100. When 25 bytes have been transferred from DSP to ARM platform, the DSP buffer descriptor is normally closed while the ARM platform buffer descriptor will have the L bit set and the byte count updated to 25.
20 R	erroR. Indicates an error occurred on the channel's buffer descriptor requested command. Some scripts may overwrite the command field with an error code indicating the source of the error. 0 No Error 1 Error
19 I	Interrupt. When SDMA has finished to process data transfer attached to this buffer descriptor, send an interrupt to the ARM platform. 0 No Interrupt 1 Interrupt the processor when BD is complete
18 C	CONTinuous. This buffer is allowed to receive multiple transmit buffers or is allowed to transmit to multiple receive buffers.The Continuos bit is decoded at the end of the processing of a BD to determine if the SDMA script must open a new BD to potentially continue the data transfer. 0 No further buffer descriptors 1 SDMA should move to the next Buffer descriptor after this one
17 W	Wrap. Indicates if this buffer descriptor is the last one for the channel control block. When encountering this bit set, the SDMA scripts updates the CurrentBD pointer to point to the first Buffer Descriptor of the array. This bit is set if the ARM platform wants to organize the array of BD in a circular way (like a ring). When all BD have been processed and if Wrap bit and CONtinuous bit are set in the last BD, the SDMA script will wrap around and it will try to re-open the first BD. 0 No Error 1 Wrap to first buffer descriptor after this one is processed.
16 D	D - "Done": bit 16: indicates the "ownership" of the buffer descriptor. When D=0 the host owns the buffer descriptor; when D=1 SDMA owns the buffer descriptor. In the case of the channel 0, D=1 indicates the SDMA has not yet processed this buffer, D=0 indicates the SDMA has processed this buffer. 0 ARM platform owns the buffer. 1 SDMA owns the buffer
15-0 Count	Count. the count field (bit 15-0) indicates the size of the data to be transmitted, the size of the data buffer pointed to by the buffer descriptor. The SDMA memory structure is different for program memory (16-bits shorts/half-words) and data memory (32-bits long). For channel 0 buffer descriptors, Count is expressed in 16-bit half-words when PM is addressed and in 32-bit words when DM is addressed. Count is typically expressed in bytes for other channel scripts, but the unit is dependant on the script.
31-0	Buffer address. Address pointer to the data buffer.
31-0	Extended buffer address. Additional pointer or other information required by some scripts.

The buffer descriptors form an array of programmable size. If the last buffer descriptor is marked by the Wrap flag-bit W=1, the array of buffer descriptor is treated as a ring with some logically continuous portion owned by the ARM platform with D=0, and the remainder owned by the SDMA with D=1. The count field of the buffer descriptor indicates how much data has been transmitted.

If ARM platform has prepared 3 buffers to be filled by the SDMA script, it has also prepared 3 BD, one for each buffer. The *Cont* and *Wrap* bits are used to organize the buffers in a circular way. For example, *CONT*inous bit is set to 1 in the 2 first BDs and *Wrap* is set in the 3rd BD. The SDMA script opens and processes BD#1. Since *CONT*inous bit is set for this BD, the SDMA will open the second BD and it will process it. Each time a BD is processed, its *Done* bit is reset by the SDMA. After the 3rd BD, if *CONT*inous is not set but if *Wrap* is set, the SDMA script stops here and the next time the channel will be triggered, the script will open the BD pointed by the currentBDptr pointer of the CCB and it will correspond to the first buffer descriptor.

If the *CONT*inous bit and *Wrap* bits are both set in the 3rd BD, the script will close it and it will try to open the first BD. An error may occur at this point if the BD#1 has already been processed and its *Done* bit is 0. The SDMA script cannot process a BD with a *Done* bit to 0. It means the BD is not ready to be processed. To avoid this situation, the *CONT*inous bit should not be set for the last BD if *Wrap* is set, and the Interrupt flag must set for the last BD. It will warn the owner of the BD that all the BDs have been processed and it has to re-set to 1 the *Done* bit of all the BD's if it desires the SDMA to fill them again. Basically, if the ARM platform expects the SDMA to fill up the buffers in a circular fashion, then it's the responsibility of the ARM platform to set the *Done* bit of a buffer descriptor at an appropriate time.

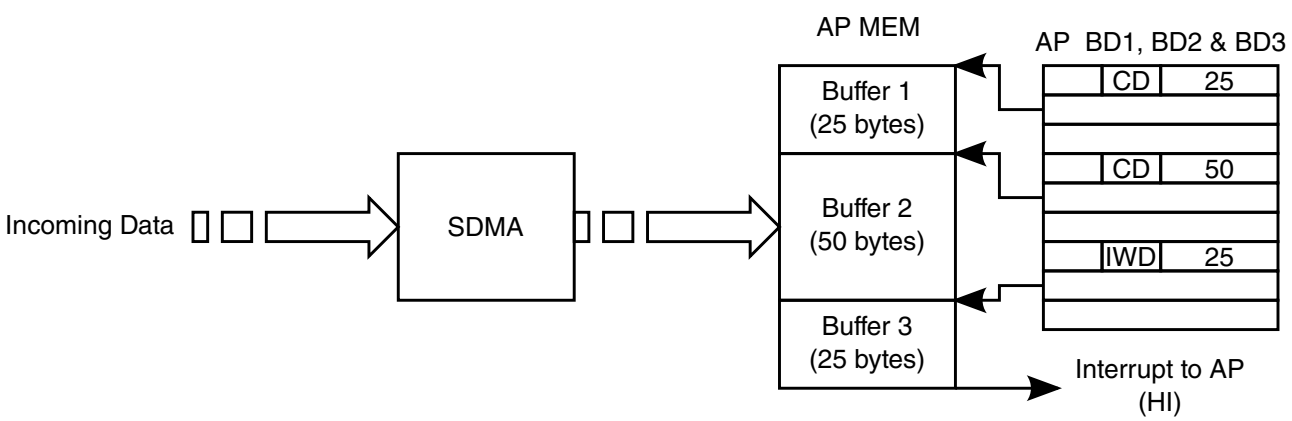


Figure 55-17. Buffer Descriptor Flow

The previous figure shows an example buffer descriptor flow. When the incoming data is stored and fills the first buffer of 25 bytes, the SDMA script opens the second BD because the CONTinuous bit was set. Then next incoming data is put in the second buffer. After receiving 50 bytes, the second buffer descriptor is also closed. The Done bit is reset and the third BD is opened. After receiving another 25 bytes, the third buffer is full and an interrupt is sent to the ARM platform because the Interrupt flag is set in the 3rd BD. The CONTinuous flag is not present the transfer is over. The next time the script will be triggered, the BD to be opened will be the first buffer descriptor since the Wrap flag was set in the 3rd BD. It is the ARM platform responsibility to set the Done bit of all the BD if it wants to use the same buffers.

55.7.1.2 Buffer Descriptor Commands for Bootload scripts

The command field of the buffer descriptor is defined separately for each script.

The following table lists the buffer descriptor commands defined for the channel 0 bootload script.

Table 55-50. Channel Zero Buffer Descriptor Commands

Command Field (binary)	Command	Description	Buffer Address	Extended Buffer Address
0000_0001 (0x01)	C0_SET_DM	Load SDMA data memory (RAM) from ARM platform memory buffer	ARM platform memory source address	SDMA memory destination address
0000_0010 (0x02)	C0_GET_DM	Copy SDMA data memory (RAM) to ARM platform memory buffer	ARM platform memory destination address	SDMA memory source address
0000_0100 (0x04)	C0_SET_PM	Load SDMA program memory (RAM) from ARM platform memory buffer	ARM platform memory source address	SDMA memory destination address
0000_0110 (0x06)	C0_GET_PM	Copy SDMA program memory (RAM) to ARM platform memory buffer	ARM platform memory destination address	SDMA memory source address
cccc_c111 (0x07 | CHN)	C0_SETCTX	Load Context for channel cccc into SDMA RAM from ARM platform memory buffer	ARM Platform memory source address	-
cccc_c011 (0x03 | CHN)	C0_GETCTXT	Copy Context for channel ccccc from SDMA RAM to ARM platform memory buffer	ARM platform memory destination address	-

The Channel 0 bootload commands are summarized as follows:

- C0_SET_[PM-DM]: load the buffer descriptor data in the SDMA local memory at the address pointed to by the "extended buffer address" field. The SDMA RAM can be seen as a Program Memory (PM, 16-bit address) or Data Memory (DM 32-bit address). When C0_SET_PM is used, the count field is expressed in "shorts" (16-bit

half words), this command can be used to download scripts. When C0_SET_DM is used, the count field is expressed in "long" (32-bit words), this command can be used to download channel contexts to the context channel area in RAM.

- C0_GET_[PM-DM]: write to the buffer descriptor's data buffer the content of the SDMA local memory from the address pointed to by the "extended buffer address" field for the length defined by the count in the buffer descriptor. C0_GET_PM is used to dump some part of the Program Memory (may be used to dump context of a channel), therefore count is expressed in "shorts"; while C0_GET_DM is used to dump to the buffer descriptor's data buffer, so the count field is in "longs."
- C0_SETCTX: load a context into the SDMA context page area. The handling script decodes the channel number from the 5 MSB of the command field of the buffer descriptor. Using the channel number the script computes the offset of the context data pointer for the channel relative to the context page base to use as the destination address in SDMA memory. Then the C0_SET_DM command explained above is invoked to load SDMA RAM from memory. The counter indicates the size in words of the context structure.
- Command value: (in binary) cccc c111, where ccccc is the channel number (5 bits). For instance, 0x0F means set context for channel 1, 0xFF means set context for channel 31.

- C0_GETCTX: write to the buffer descriptor's data buffer the content of the SDMA context page area. The handling script decodes the channel number from the 5 MSB of the command field of the buffer descriptor. Using this channel number, the script computes the offset of the context data pointer for the channel relative to the context page base to use as the source address for the copy. Then the C0_GET_DM command explained above is invoked to copy the context to memory. The counter indicates the size in words of the context structure.
- Command value: (in binary): cccc c011, where ccccc is the channel number (5 bits). For instance, 0x03 means get context of channel 1, 0xFB means get context of channel 31.

NOTE

To download channel context, C0_SETDM and C0_SETCTXT command can be used but the second one is easier because the channel number is embedded into the command field, whereas with the C0_SETDM, the pointer to the channel context area must be written into the extended buffer address field of the buffer descriptor.

55.7.1.3 Example of Buffer Descriptors for Channel 0.

Figure 55-19 illustrates the buffer descriptors that must be set in ARM platform memory space, before execution of boot code, to download contexts and scripts of channels 1, 4, and 10. After boot code execution, SDMA memory will be populated with the different contexts and scripts as presented in the following figure.

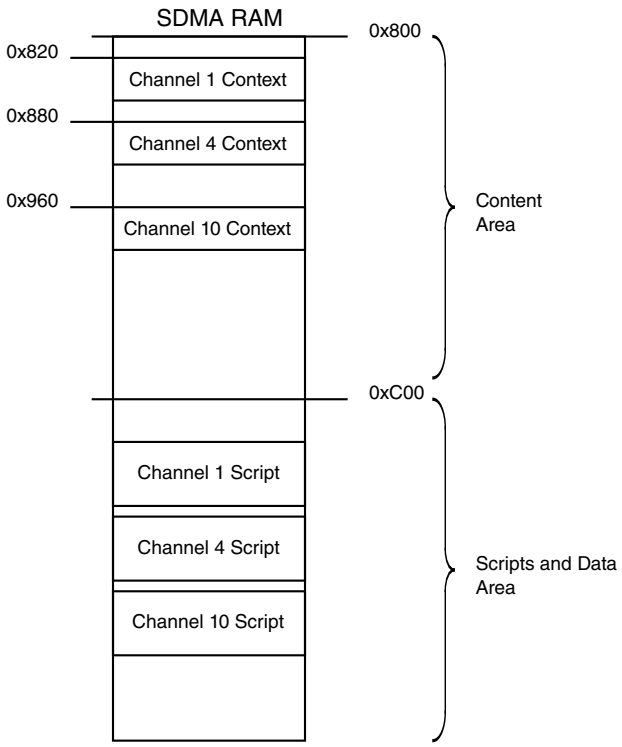
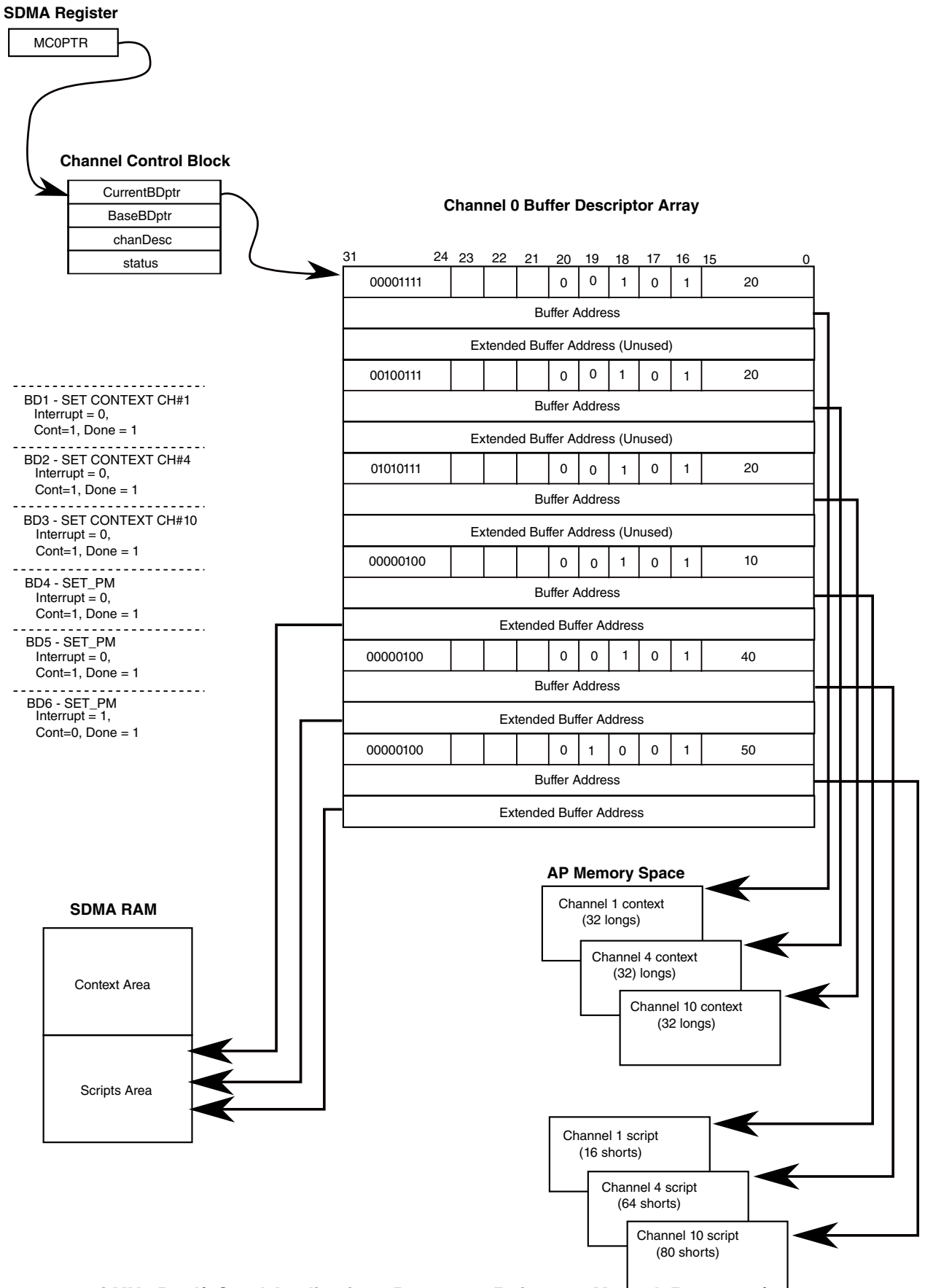


Figure 55-18. Example of SDMA RAM After Boot Session


SDMA Register
MC0PTR
Channel Control Block
CurrentBDptr
BaseBDptr
chanDesc
status
Channel 0 Buffer Descriptor Array
31 24 23 22 21 20 19 18 17 16 15 0
00001111 0 0 1 0 1 20
Buffer Address
Extended Buffer Address (Unused)
00100111 0 0 1 0 1 20
Buffer Address
Extended Buffer Address (Unused)
01010111 0 0 1 0 1 20
Buffer Address
Extended Buffer Address (Unused)
00000100 0 0 1 0 1 10
Buffer Address
Extended Buffer Address
00000100 0 0 1 0 1 40
Buffer Address
Extended Buffer Address
00000100 0 1 0 0 1 50
Buffer Address
Extended Buffer Address
BD1 - SET CONTEXT CH#1
Interrupt = 0,
Cont=1, Done = 1
BD2 - SET CONTEXT CH#4
Interrupt = 0,
Cont=1, Done = 1
BD3 - SET CONTEXT CH#10
Interrupt = 0,
Cont=1, Done = 1
BD4 - SET_PM
Interrupt = 0,
Cont=1, Done = 1
BD5 - SET_PM
Interrupt = 0,
Cont=1, Done = 1
BD6 - SET_PM
Interrupt = 1,
Cont=0, Done = 1
SDMA RAM
Context Area
Scripts Area
AP Memory Space
Channel 1 context
(32 longs)
Channel 4 context
(32) longs)
Channel 10 context
(32 longs)
Channel 1 script
(16 shorts)
Channel 4 script
(64 shorts)
Channel 10 script
(80 shorts)

55.7.1.4 Channel Context

There are 32 channel context memory structures pointed to by the local save area pointer. These channel context memory structures are fixed.

The script in the SDMA computes the memory offset for a given channel based on the structure length and channel number. Figure below shows the structure of the channel context as it is saved in the SDMA local memory (RAM).

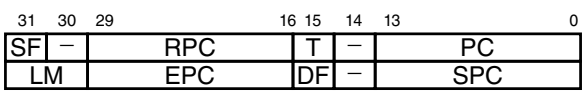
A channel context consists in 24 words, one per register. A total of 32 words are reserved for every channel. The additional 8 words are called scratch ram and they are dedicated to each channel. This memory area is commonly used for stack management.

The structure is divided in 4 areas:

- Channel status registers
- General purpose registers
- Functional units state registers reflecting the state of the ARM platform DMAs (Burst and Peripheral DMA).
- Scratch RAM

The details of the channel context status registers are described in the following figure.

The PC field of the first long register must point to the SDMA RAM address where the script that will be executed on the channel is located and this value equals the one stored in the extended buffer address of the buffer descriptor with C0_SETPM command.



SF: Source fault while loading data
RPC: Return program counter
T: Test bit: status of arithmetic and test instructions
PC: Program counter
LM: Loop mode
EPC: Loop end program counter
DF: Destination fault while storing data
SPC: Loop Start program counter

Figure 55-20. SDMA State Registers (ShPC, ShLoop)

55.7.2 Typical Data Transfer Supported by SDMA DMA Units

This section presents a library of SDMA scripts that perform data transfers through the peripheral DMA and the burst DMA units.

The ARM platform memory and peripherals are devices that either the peripheral DMA or the burst DMA can access. The scripts are given for a peripheral DMA whose address registers are programmed in incremented mode when internal memory is involved. See the following table for the summary.

Table 55-51. Typical Data Transfers Summary

Data Transfer	Peripheral DMA	Burst DMA	Comments
ARM platform External Memory ↔ ARM platform External Memory		3	Copy mode Script example, see Burst DMA Unit Copy Mode and External Memory to External Memory.
ARM platform Peripheral ↔ ARM platform Peripheral	3		Copy mode if same data path width Script example, see Peripheral to Peripheral Transfer.
ARM platform External Memory ↔ ARM platform Peripheral	3	3	Data transit through SDMA Script example, see Transfer Between Peripheral and External Memory.
ARM platform External Memory ↔ ARM platform Internal Memory		3	Copy mode Script example, see Transfer Between External Memory and Internal Memory.
ARM platform Internal Memory ↔ ARM platform Internal Memory		3	Copy mode Script example, see Internal Memory to Internal Memory.
ARM platform Internal memory ↔ ARM platform Peripheral	3		Data transit through SDMA Script example, see Transfer Between Peripheral and Internal Memory.

NOTE

These scripts are provided as examples of how to use DMA blocks to perform required data transfers: They are not "official" programs.

55.7.2.1 External Memory to External Memory

This section describes the SDMA script that performs data moves in external memory.

For this particular data transfer, only the burst DMA is used. It is programmed in copy mode, so no data transmits through an SDMA general register.

The SDMA core only monitors data transfer status. It is assumed source and destination address values are already present in two SDMA general registers (r1 and r2). For this example, it is also assumed that a 32-bit word-to-move for source-to-destination address is present in r0 and equals 64.

Data Moves in External Memory

```
1         stf r1,MSA                          // Source address setup

2         stf r2,MDA                          // Destination address setup

3         ldi r0,0x64                         // 64 words must be transferred from MSA to
MDA

4         ldi r1,0x8

MAIN_XFER:

5         cmphs r0,r1                         // Is r0 >= 0x8

6         bf LAST_XFER                      // If not, jump to last transfer label

7         stf r1,MD|CPY                        // Copy 8 words from MSA to MDA address.

8         subi r0,0x8                         // Decrement counter

9         jmp MAIN_XFER                     // return to main transfer loop

LAST_XFER:

10        stf r0,MD|CPY                        // perform last transfer
```

All instructions are performed in one cycle (jumps excepted). Instruction 7 triggers a copy transfer: A read burst access of 8-word starts, data is staged in MD and then a write burst of 8 words is executed. Instruction 8, 9, 5, and 6 are executed while the burst access is in progress. If this access is not complete when instruction 7 is executed a second time, SDMA stalls on this instruction as long as the previous copy transfer is not over. In this case, the instruction is no longer a one-cycle instruction.

During the main loop (MAIN_XFER), r1 always equals 8, so burst lengths are 8 words. On the last ldf |CPY instruction (10), r1 equals the reminder of r0 divided by 8; therefore, the length of bursts triggered in copy mode equal r1 value, which is between 1 and 7.

55.7.2.2 Peripheral to Peripheral Transfer

For this data transfer, only the peripheral DMA is used.

It is programmed in copy mode, so no data will transmit through the SDMA general register used in the ldf instruction, but the contents of the general register are lost. The SDMA core only monitors the transfer.

55.7.2.2.1 Source and Destination Target Have the Same Data Path Width

When the source and destination target have the same data path width, the following is true:

- Source target is a *half-word* (16-bit) peripheral located at address 0x1002.
- Destination is a *half-word* (16-bit) peripheral located at address 0x2006.

It is assumed the address values are already present in two SDMA general registers (r1, r2). The script for a transfer of 10 half-word is as follows:

Same Data Path Width for Source and Destination

```
//SETUP SECTION
1         stf r1, PSA|SZ16|F                    //r1=0x1002 Source address register setup
2         stf r2, PDA|SZ16|F                    //r2=0x2006 Destination address register
setup
3         bdf ERROR_ADDR_SETUP
4         ldi r0,0xa                           //loop counter is 10
//MAIN LOOP TRANFER
copy_loop:

5         loop 2,0
6           ldf r7,PD|CPY                      //Reads 1 half-word from src and writes to
dest.
7           yield
8         bdf ERROR_DURING_XFER
ERROR_ADDR_SETUP:
                    //correction of PSA/PDA setup and jumps to main loop transfer
ERROR_DURING_XFER:
//flag error is set,
//PS can be read to know if error occurs during read or write access.
```

If a data transfer must occur between two word peripherals, only the setup section should be updated. The transfer itself is always performed by the hardware loop instruction.

All instructions are executed in one cycle (change of flow excepted). On instruction 6, a single read access is triggered, read data is staged in PD, and a write-to-destination is executed. When the transfers are in progress, the SDMA can execute he next instructions in parallel. If instruction 6, which performs the copy transfer, is executed while the previous access is not over, SDMA is stalled and instruction ldf is a multi-cycle instruction.

55.7.2.2.2 Source and Destination Target Have a Different Data Path Width

When the source and destination target have a different data path width, copy mode cannot be used, and any attempt to initiate a copy transfer immediately raises an error, which is stored in the SF flag.

The following example shows the SDMA code that could transfer 10 words from a *word* (32-bit) peripheral to a *half-word* peripheral whose addresses are preliminary and stored in r1 and r2.

Different Data Path Width for Source and Destination

```
//SETUP SECTION
1          stf r1, PSA|SZ32|F|PF                    //r1=0x1000 and prefetch data
2          stf r2, PDA|SZ16|F                      //r2=0x2006
3          bdf ERROR_ADDR_SETUP
4          ldi r0,0xa                             //loop counter is 10
//MAIN LOOP TRANFER
main_loop_xfer_16_16:
5          loop 6,0
6            ldf r7,PD                           //copy 32-bit of PD in r7
7            stf r7,PD                            //store 16 LSB of r7 in PD and a flush is
executed
8            rorb r7
9            rorb r7                            //16 MSB --> 16 LSB
10           stf r7,PD                            //store 16 LSB of r6 in PD and a flush.
11           yield
```

On instruction 1, when the source address register is programmed and a data prefetch is required, a read access is executed. In parallel, the SDMA executes instructions 2 to 5. On instruction 6, the SDMA tries to read data that was fetched by instruction 1. If data is ready, the ldf will be a one cycle instruction; otherwise, the SDMA is stalled as long as the read access is not finished. Then, the 16 LSB of the read data is stored in PD and automatically flushed to the destination peripheral. In parallel, the SDMA executes the rotation instructions (8, 9), and stores the 16 MSB of the read data into PD. If a previous write access is finished, instruction 10 will be a one-cycle instruction.

The main loop transfer may appear inefficient, but due to wait states imposed to the peripheral DMA each time an external access is performed, a software pipeline is in place. During the time needed to flush PD, the SDMA executes the move and rotation operations. SDMA executes instructions in parallel with DMA accesses.

55.7.2.3 Transfer Between Peripheral and External Memory

55.7.2.3.1 Peripheral to External Memory Transfer

A transfer from a peripheral to the external memory controller involves the peripheral DMA and the burst DMA.

The code for transferring 100 word from word peripheral to the external memory would be as follows:

Peripheral to External Memory Transfer

```
//SETUP SECTION source and destination addresses are already in r1 and r2
1          stf r1, PSA|SZ16|F|PF                     //r1=0x1000 and prefetch 32-bit data
2          stf r2, MDA                             //r2=0x2000, setup burst DMA destination
address
3          bdf ERROR_ADDR_SETUP
4          ldi r0,0x64                             //loop counter is 100
5
           //MAIN LOOP TRANFER
6          loop 3,0
```

```
7           ldf r1,PD|PF                    // read 32 bits of PD and initiate a new read
access.
8           stf r1,MD|32                     // store 32 bits of r1 in the MD fifo.
9           yield
10        ldf r1,PD                        // last word data is read
11        stf r1,MD|32|FL                    // to flush all remaining bytes of MD
```

On instruction 1, the source address register of the peripheral DMA is programmed and data is fetched. This data is stored in PD and the SDMA reads PD during instruction 7, which is a one-cycle instruction that is read-access finished. On the same instruction (7), a data prefetch is required and a read access to the source peripheral is executed. In parallel, the SDMA stored the previous read data into the data register of MD. When MD (which is an eight-word FIFO) is full, a burst write access is executed to empty the FIFO. As long as the next SDMA instructions do not access the burst DMA, they will be one-cycle instructions. The following figures show how the peripheral DMA and burst DMA work in parallel.

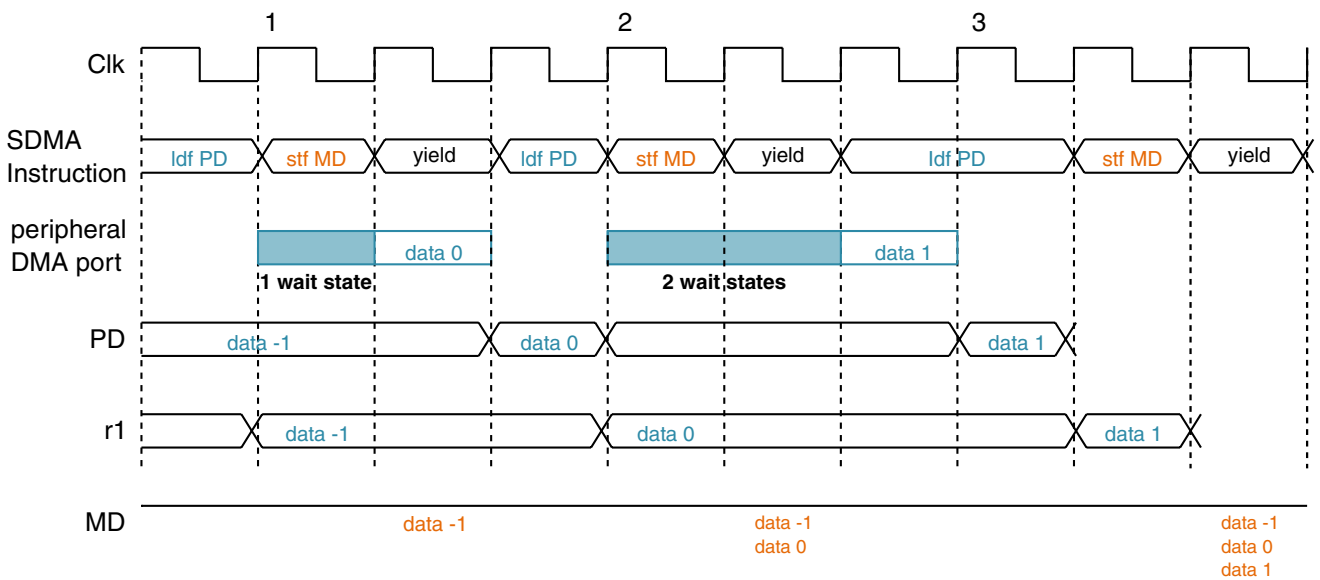


Figure 55-21. Peripheral to External Memory Example (1)

As seen in the figure above, the read access triggered by the ldf PD instruction is symbolized by the blue bar when in progress. After wait states, the read data (data 0, data 1) is stored in PD on the clk rising edge. On edge 2, data 0 is available in PD so it can be transferred to the SDMA general register r1, and then stored in MD FIFO. On edge 3, data 1 is not in PD; therefore, SDMA is stalled on the ldf instruction, which lasts two cycles. The figure below shows an example of when MD FIFO is full with data.

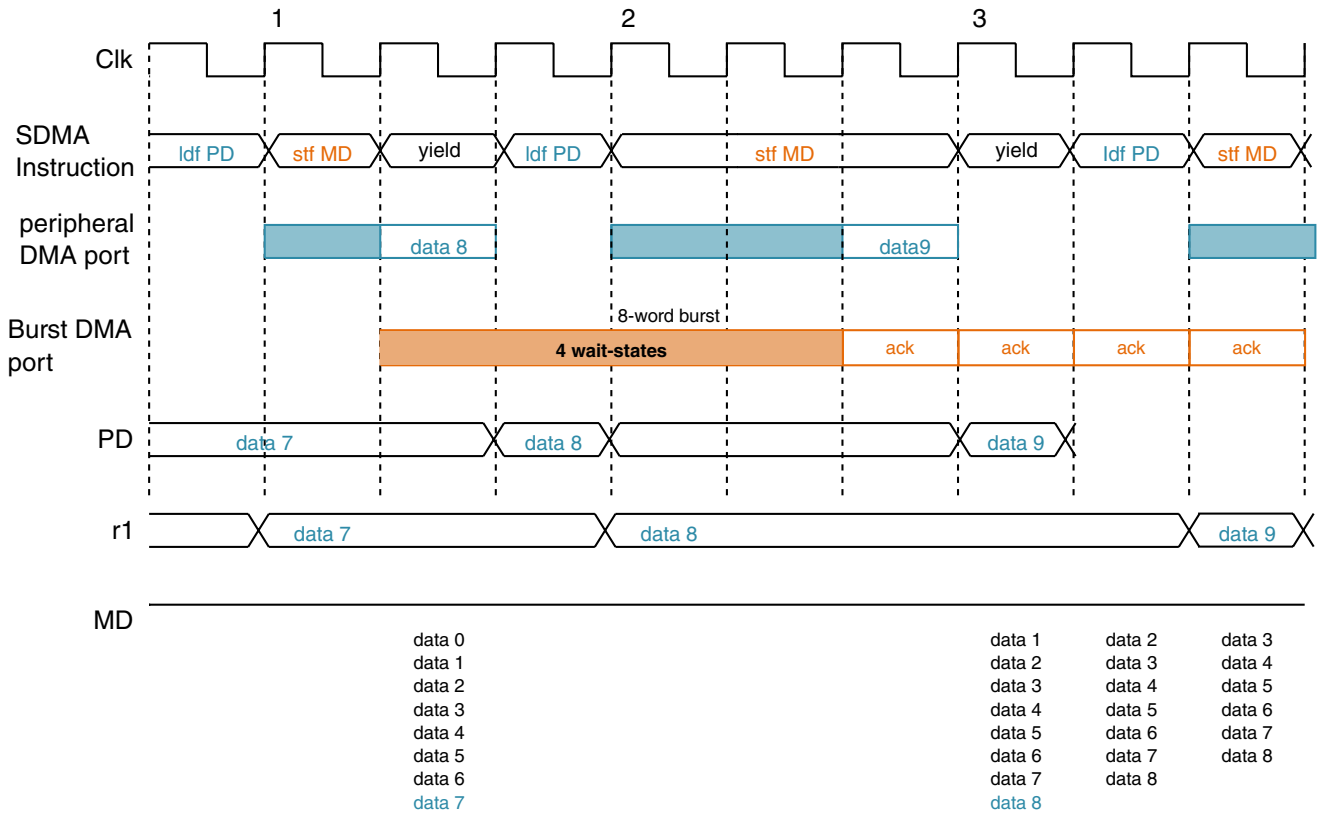


Figure 55-22. Peripheral to External Memory Example (2)

In the previous figure, the write bar means the burst DMA is performing a write burst access. The latency to have the first write acknowledge is four cycles. SDMA is stalled on instruction stf because no acknowledge was received, MD FIFO is full, and there is no empty slot to store data 9. When an acknowledge is sampled by the burst DMA, FIFO is shifted and data 8 is written. As long as there is at least one empty slot in MD FIFO, the stf MD instruction lasts one cycle.

55.7.2.3.2 External Memory to Peripheral Transfer

A transfer from the external memory to a peripheral involves the peripheral DMA and the burst DMA.

The code for transferring 100 word from external memory to a word peripheral would be as follows:

External Memory to Peripheral Transfer

```
//SETUP SECTION source and destination addresses are already in r1 and r2
1         stf r1, MSA|PF              //r1=0x1000 and starts a 8-word read burst
2         stf r2, PDA|SZ32|P           //r2=0x2010, setup peripheral DMA destination address
3         bdf ERROR_ADDR_SETUP
4         ldi r0,0x64                //loop counter is 100
//MAIN LOOP TRANFER
6         loop 3,0
```

```
7           ldf r1,MD|32|PF          // read 32 bits of MD and initiate a new read access
                                  // if MD is empty after this reading.
8           stf r1,PD               // store 32 bits of r1 in the PD.
9           yield
          10 ldf r1,MD|32            // last word data is read
11        stf r1,PD                 // last write access
```

On instruction 1, a read burst of 8 words begins. Read data is staged into MD. On instruction 7 (and if data is available in MD), 32 bits are copied into r1. Then instruction 8 writes them into PD and an automatic flush is executed. The SDMA core, peripheral DMA, and burst DMA can work in parallel as long as no SDMA instruction tries to start a new write access on the peripheral DMA while the previous access is still in progress, or as long as there is data in MD when the SDMA tries to read it.

55.7.2.4 Transfer Between External Memory and Internal Memory

Since the internal memory (ARM platform RAM) is accessed via the peripheral DMA and the external memory is accessed via the burst DMA, the SDMA scripts that are described in Transfer Between Peripheral and External Memory can be reused. The exception is that the peripheral DMA address registers (PSA or PDA, depending on the script) should be programmed in incremented mode rather than frozen mode.

55.7.2.4.1 Internal Memory to Internal Memory

The internal memory can only be accessed via the peripheral DMA, so the script described in Peripheral to Peripheral Transfer can be reused with a different programming of the peripheral DMA address registers.

55.7.2.4.2 Transfer Between Peripheral and Internal Memory

For this transfer, the peripheral DMA is also used in copy mode.

The SDMA script is very similar to the one described in Peripheral to Peripheral Transfer, except for the peripheral DMA address registers programming.

55.8 ARM Platform Memory Map and Control Register Definitions

The ARM platform controls the SDMA by means of several interface registers. Those registers are described in the current section.

All registers are clocked with the SDMA clock (which means the ARM platform must ensure that the SDMA clock is running when it wants to access any register).

SDMAARM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20E_C000	ARM platform Channel 0 Pointer (SDMAARM_MC0PTR)	32	R/W	0000_0000h	55.8.1/4894
20E_C004	Channel Interrupts (SDMAARM_INTR)	32	w1c	0000_0000h	55.8.2/4894
20E_C008	Channel Stop/Channel Status (SDMAARM_STOP_STAT)	32	w1c	0000_0000h	55.8.3/4894
20E_C00C	Channel Start (SDMAARM_HSTART)	32	R/W	0000_0000h	55.8.4/4895
20E_C010	Channel Event Override (SDMAARM_EVTOVR)	32	R/W	0000_0000h	55.8.5/4895
20E_C014	Channel BP Override (SDMAARM_DSPOVR)	32	R/W	FFFF_FFFFh	55.8.6/4896
20E_C018	Channel ARM platform Override (SDMAARM_HOSTOVR)	32	R/W	0000_0000h	55.8.7/4896
20E_C01C	Channel Event Pending (SDMAARM_EVTPEND)	32	w1c	0000_0000h	55.8.8/4896
20E_C024	Reset Register (SDMAARM_RESET)	32	R	0000_0000h	55.8.9/4897
20E_C028	DMA Request Error Register (SDMAARM_EVTERR)	32	R	0000_0000h	55.8.10/ 4898
20E_C02C	Channel ARM platform Interrupt Mask (SDMAARM_INTRMASK)	32	R/W	0000_0000h	55.8.11/ 4898
20E_C030	Schedule Status (SDMAARM_PSW)	32	R	0000_0000h	55.8.12/ 4899
20E_C034	DMA Request Error Register (SDMAARM_EVTERRDBG)	32	R	0000_0000h	55.8.13/ 4899
20E_C038	Configuration Register (SDMAARM_CONFIG)	32	R/W	0000_0003h	55.8.14/ 4900
20E_C03C	SDMA LOCK (SDMAARM_SDMA_LOCK)	32	R/W	0000_0000h	55.8.15/ 4901
20E_C040	OnCE Enable (SDMAARM_ONCE_ENB)	32	R/W	0000_0000h	55.8.16/ 4902
20E_C044	OnCE Data Register (SDMAARM_ONCE_DATA)	32	R/W	0000_0000h	55.8.17/ 4903
20E_C048	OnCE Instruction Register (SDMAARM_ONCE_INSTR)	32	R/W	0000_0000h	55.8.18/ 4903
20E_C04C	OnCE Status Register (SDMAARM_ONCE_STAT)	32	R	0000_E000h	55.8.19/ 4903
20E_C050	OnCE Command Register (SDMAARM_ONCE_CMD)	32	R/W	0000_0000h	55.8.20/ 4905

Table continues on the next page...

SDMAARM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20E_C058	Illegal Instruction Trap Address (SDMAARM_ILLINSTADDR)	32	R/W	0000_0001h	55.8.21/ 4906
20E_C05C	Channel 0 Boot Address (SDMAARM_CHN0ADDR)	32	R/W	0000_0050h	55.8.22/ 4906
20E_C060	DMA Requests (SDMAARM_EVT_MIRROR)	32	R	0000_0000h	55.8.23/ 4907
20E_C064	DMA Requests 2 (SDMAARM_EVT_MIRROR2)	32	R	0000_0000h	55.8.24/ 4907
20E_C070	Cross-Trigger Events Configuration Register 1 (SDMAARM_XTRIG_CONF1)	32	R/W	0000_0000h	55.8.25/ 4908
20E_C074	Cross-Trigger Events Configuration Register 2 (SDMAARM_XTRIG_CONF2)	32	R/W	0000_0000h	55.8.26/ 4910
20E_C100	Channel Priority Registers (SDMAARM_SDMA_CHNPRI0)	32	R/W	0000_0000h	55.8.27/ 4911
20E_C104	Channel Priority Registers (SDMAARM_SDMA_CHNPRI1)	32	R/W	0000_0000h	55.8.27/ 4911
20E_C108	Channel Priority Registers (SDMAARM_SDMA_CHNPRI2)	32	R/W	0000_0000h	55.8.27/ 4911
20E_C10C	Channel Priority Registers (SDMAARM_SDMA_CHNPRI3)	32	R/W	0000_0000h	55.8.27/ 4911
20E_C110	Channel Priority Registers (SDMAARM_SDMA_CHNPRI4)	32	R/W	0000_0000h	55.8.27/ 4911
20E_C114	Channel Priority Registers (SDMAARM_SDMA_CHNPRI5)	32	R/W	0000_0000h	55.8.27/ 4911
20E_C118	Channel Priority Registers (SDMAARM_SDMA_CHNPRI6)	32	R/W	0000_0000h	55.8.27/ 4911
20E_C11C	Channel Priority Registers (SDMAARM_SDMA_CHNPRI7)	32	R/W	0000_0000h	55.8.27/ 4911
20E_C120	Channel Priority Registers (SDMAARM_SDMA_CHNPRI8)	32	R/W	0000_0000h	55.8.27/ 4911
20E_C124	Channel Priority Registers (SDMAARM_SDMA_CHNPRI9)	32	R/W	0000_0000h	55.8.27/ 4911
20E_C128	Channel Priority Registers (SDMAARM_SDMA_CHNPRI10)	32	R/W	0000_0000h	55.8.27/ 4911
20E_C12C	Channel Priority Registers (SDMAARM_SDMA_CHNPRI11)	32	R/W	0000_0000h	55.8.27/ 4911
20E_C130	Channel Priority Registers (SDMAARM_SDMA_CHNPRI12)	32	R/W	0000_0000h	55.8.27/ 4911
20E_C134	Channel Priority Registers (SDMAARM_SDMA_CHNPRI13)	32	R/W	0000_0000h	55.8.27/ 4911
20E_C138	Channel Priority Registers (SDMAARM_SDMA_CHNPRI14)	32	R/W	0000_0000h	55.8.27/ 4911
20E_C13C	Channel Priority Registers (SDMAARM_SDMA_CHNPRI15)	32	R/W	0000_0000h	55.8.27/ 4911

Table continues on the next page...

SDMAARM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20E_C140	Channel Priority Registers (SDMAARM_SDMA_CHNPRI16)	32	R/W	0000_0000h	55.8.27/ 4911
20E_C144	Channel Priority Registers (SDMAARM_SDMA_CHNPRI17)	32	R/W	0000_0000h	55.8.27/ 4911
20E_C148	Channel Priority Registers (SDMAARM_SDMA_CHNPRI18)	32	R/W	0000_0000h	55.8.27/ 4911
20E_C14C	Channel Priority Registers (SDMAARM_SDMA_CHNPRI19)	32	R/W	0000_0000h	55.8.27/ 4911
20E_C150	Channel Priority Registers (SDMAARM_SDMA_CHNPRI20)	32	R/W	0000_0000h	55.8.27/ 4911
20E_C154	Channel Priority Registers (SDMAARM_SDMA_CHNPRI21)	32	R/W	0000_0000h	55.8.27/ 4911
20E_C158	Channel Priority Registers (SDMAARM_SDMA_CHNPRI22)	32	R/W	0000_0000h	55.8.27/ 4911
20E_C15C	Channel Priority Registers (SDMAARM_SDMA_CHNPRI23)	32	R/W	0000_0000h	55.8.27/ 4911
20E_C160	Channel Priority Registers (SDMAARM_SDMA_CHNPRI24)	32	R/W	0000_0000h	55.8.27/ 4911
20E_C164	Channel Priority Registers (SDMAARM_SDMA_CHNPRI25)	32	R/W	0000_0000h	55.8.27/ 4911
20E_C168	Channel Priority Registers (SDMAARM_SDMA_CHNPRI26)	32	R/W	0000_0000h	55.8.27/ 4911
20E_C16C	Channel Priority Registers (SDMAARM_SDMA_CHNPRI27)	32	R/W	0000_0000h	55.8.27/ 4911
20E_C170	Channel Priority Registers (SDMAARM_SDMA_CHNPRI28)	32	R/W	0000_0000h	55.8.27/ 4911
20E_C174	Channel Priority Registers (SDMAARM_SDMA_CHNPRI29)	32	R/W	0000_0000h	55.8.27/ 4911
20E_C178	Channel Priority Registers (SDMAARM_SDMA_CHNPRI30)	32	R/W	0000_0000h	55.8.27/ 4911
20E_C17C	Channel Priority Registers (SDMAARM_SDMA_CHNPRI31)	32	R/W	0000_0000h	55.8.27/ 4911
20E_C200	Channel Enable RAM (SDMAARM_CHNENBL0)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C204	Channel Enable RAM (SDMAARM_CHNENBL1)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C208	Channel Enable RAM (SDMAARM_CHNENBL2)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C20C	Channel Enable RAM (SDMAARM_CHNENBL3)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C210	Channel Enable RAM (SDMAARM_CHNENBL4)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C214	Channel Enable RAM (SDMAARM_CHNENBL5)	32	R/W	0000_0000h	55.8.28/ 4911

Table continues on the next page...

SDMAARM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20E_C218	Channel Enable RAM (SDMAARM_CHNENBL6)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C21C	Channel Enable RAM (SDMAARM_CHNENBL7)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C220	Channel Enable RAM (SDMAARM_CHNENBL8)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C224	Channel Enable RAM (SDMAARM_CHNENBL9)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C228	Channel Enable RAM (SDMAARM_CHNENBL10)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C22C	Channel Enable RAM (SDMAARM_CHNENBL11)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C230	Channel Enable RAM (SDMAARM_CHNENBL12)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C234	Channel Enable RAM (SDMAARM_CHNENBL13)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C238	Channel Enable RAM (SDMAARM_CHNENBL14)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C23C	Channel Enable RAM (SDMAARM_CHNENBL15)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C240	Channel Enable RAM (SDMAARM_CHNENBL16)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C244	Channel Enable RAM (SDMAARM_CHNENBL17)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C248	Channel Enable RAM (SDMAARM_CHNENBL18)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C24C	Channel Enable RAM (SDMAARM_CHNENBL19)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C250	Channel Enable RAM (SDMAARM_CHNENBL20)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C254	Channel Enable RAM (SDMAARM_CHNENBL21)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C258	Channel Enable RAM (SDMAARM_CHNENBL22)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C25C	Channel Enable RAM (SDMAARM_CHNENBL23)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C260	Channel Enable RAM (SDMAARM_CHNENBL24)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C264	Channel Enable RAM (SDMAARM_CHNENBL25)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C268	Channel Enable RAM (SDMAARM_CHNENBL26)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C26C	Channel Enable RAM (SDMAARM_CHNENBL27)	32	R/W	0000_0000h	55.8.28/ 4911

Table continues on the next page...

SDMAARM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20E_C270	Channel Enable RAM (SDMAARM_CHNENBL28)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C274	Channel Enable RAM (SDMAARM_CHNENBL29)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C278	Channel Enable RAM (SDMAARM_CHNENBL30)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C27C	Channel Enable RAM (SDMAARM_CHNENBL31)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C280	Channel Enable RAM (SDMAARM_CHNENBL32)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C284	Channel Enable RAM (SDMAARM_CHNENBL33)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C288	Channel Enable RAM (SDMAARM_CHNENBL34)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C28C	Channel Enable RAM (SDMAARM_CHNENBL35)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C290	Channel Enable RAM (SDMAARM_CHNENBL36)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C294	Channel Enable RAM (SDMAARM_CHNENBL37)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C298	Channel Enable RAM (SDMAARM_CHNENBL38)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C29C	Channel Enable RAM (SDMAARM_CHNENBL39)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C2A0	Channel Enable RAM (SDMAARM_CHNENBL40)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C2A4	Channel Enable RAM (SDMAARM_CHNENBL41)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C2A8	Channel Enable RAM (SDMAARM_CHNENBL42)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C2AC	Channel Enable RAM (SDMAARM_CHNENBL43)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C2B0	Channel Enable RAM (SDMAARM_CHNENBL44)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C2B4	Channel Enable RAM (SDMAARM_CHNENBL45)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C2B8	Channel Enable RAM (SDMAARM_CHNENBL46)	32	R/W	0000_0000h	55.8.28/ 4911
20E_C2BC	Channel Enable RAM (SDMAARM_CHNENBL47)	32	R/W	0000_0000h	55.8.28/ 4911

55.8.1 ARM platform Channel 0 Pointer (SDMAARM_MC0PTR)

Address: 20E_C000h base + 0h offset = 20E_C000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MC0PTR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_MC0PTR field descriptions

Field	Description
MC0PTR	**Channel 0 Pointer** contains the 32-bit address, in ARM platform memory, of channel 0 control block (the boot channel). Appendix A fully describes the SDMA Application Programming Interface (API). The ARM platform has a read/write access and the SDMA has a read-only access.

55.8.2 Channel Interrupts (SDMAARM_INTR)

Address: 20E_C000h base + 4h offset = 20E_C004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HI[31:0]																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_INTR field descriptions

Field	Description
HI[31:0]	The ARM platform Interrupts register contains the 32 HI[i] bits. If any bit is set, it will cause an interrupt to the ARM platform. This register is a "write-ones" register to the ARM platform. When the ARM platform sets a bit in this register the corresponding HI[i] bit is cleared. The interrupt service routine should clear individual channel bits when their interrupts are serviced, failure to do so will cause continuous interrupts. The SDMA is responsible for setting the HI[i] bit corresponding to the current channel when the corresponding `done` instruction is executed.

55.8.3 Channel Stop/Channel Status (SDMAARM_STOP_STAT)

Address: 20E_C000h base + 8h offset = 20E_C008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HE																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_STOP_STAT field descriptions

Field	Description
HE	This 32-bit register gives access to the ARM platform Enable bits. There is one bit for every channel. This register is a "write-ones" register to the ARM platform. When the ARM platform writes 1 in bit i of this register, it clears the HE[i] and HSTART[i] bits. Reading this register yields the current state of the HE[i] bits.

55.8.4 Channel Start (SDMAARM_HSTART)

Address: 20E_C000h base + Ch offset = 20E_C00Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HSTART_HE																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_HSTART field descriptions

Field	Description
HSTART_HE	The HSTART_HE registers are 32 bits wide with one bit for every channel. When a bit is written to 1, it enables the corresponding channel. Two physical registers are accessed with that address (HSTART and HE), which enables the ARM platform to trigger a channel a second time before the first trigger is processed. • This register is a "write-ones" register to the ARM platform. Neither HSTART[i] bit can be set while the corresponding HE[i] bit is cleared. • When the ARM platform tries to set the HSTART[i] bit by writing a one (if the corresponding HE[i] bit is clear), the bit in the HSTART[i] register will remain cleared and the HE[i] bit will be set. • If the corresponding HE[i] bit was already set, the HSTART[i] bit will be set. The next time the SDMA channel *i* attempts to clear the HE[i] bit by means of a `done` instruction, the bit in the HSTART[i] register will be cleared and the HE[i] bit will take the old value of the HSTART[i] bit. • Reading this register yields the current state of the HSTART[i] bits. This mechanism enables the ARM platform to pipeline two HSTART commands per channel.

55.8.5 Channel Event Override (SDMAARM_EVTOVR)

Address: 20E_C000h base + 10h offset = 20E_C010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	EO																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_EVTOVR field descriptions

Field	Description
EO	The Channel Event Override register contains the 32 EO[i] bits. A bit set in this register causes the SDMA to ignore DMA requests when scheduling the corresponding channel.

55.8.6 Channel BP Override (SDMAARM_DSPOVR)

Address: 20E_C000h base + 14h offset = 20E_C014h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DO																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

SDMAARM_DSPOVR field descriptions

Field	Description
DO	This register is reserved. All DO bits should be set to the reset value of 1. A setting of 0 will prevent SDMA channels from starting according to the condition described in Runnable Channels Evaluation. 0 - Reserved 1 - Reset value.

55.8.7 Channel ARM platform Override (SDMAARM_HOSTOVR)

Address: 20E_C000h base + 18h offset = 20E_C018h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	HO																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_HOSTOVR field descriptions

Field	Description
HO	The Channel ARM platform Override register contains the 32 HO[i] bits. A bit set in this register causes the SDMA to ignore the ARM platform enable bit (HE) when scheduling the corresponding channel.

55.8.8 Channel Event Pending (SDMAARM_EVTPEND)

Address: 20E_C000h base + 1Ch offset = 20E_C01Ch

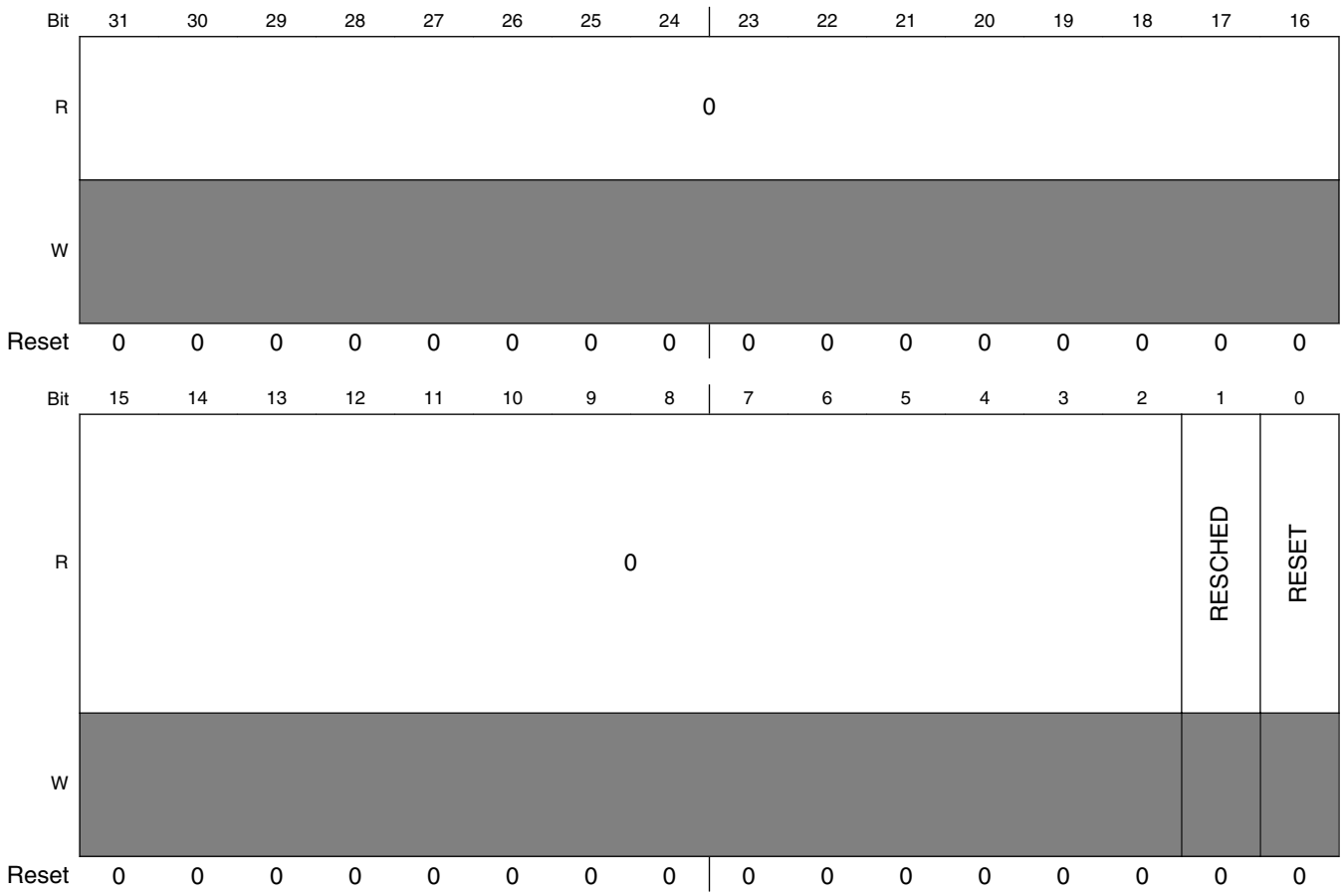
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EP																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_EVTPEND field descriptions

Field	Description
EP	The Channel Event Pending register contains the 32 EP[i] bits. Reading this register enables the ARM platform to determine what channels are pending after the reception of a DMA request. • Setting a bit in this register causes the SDMA to reevaluate scheduling as if a DMA request mapped on this channel had occurred. This is useful for starting up channels, so that initialization is done before awaiting the first request. The scheduler can also set bits in the EVTPEND register according to the received DMA requests. • The EP[i] bit may be cleared by the `done` instruction when running the channel *i* script. This a "write-ones" mechanism: Writing a '0' does not clear the corresponding bit.

55.8.9 Reset Register (SDMAARM_RESET)

Address: 20E_C000h base + 24h offset = 20E_C024h



SDMAARM_RESET field descriptions

Field	Description
31–2 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

SDMAARM_RESET field descriptions (continued)

Field	Description
1 RESCHED	When set, this bit forces the SDMA to reschedule as if a script had executed a `done` instruction. This enables the ARM platform to recover from a runaway script on a channel by clearing its HE[i] bit via the STOP register, and then forcing a reschedule via the RESCHED bit. The RESCHED bit is cleared when the context switch starts.
0 RESET	When set, this bit causes the SDMA to be held in a software reset. The internal reset signal is held low 16 cycles; the RESET bit is automatically cleared when the internal reset signal rises.

55.8.10 DMA Request Error Register (SDMAARM_EVTERR)

Address: 20E_C000h base + 28h offset = 20E_C028h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CHNERR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_EVTERR field descriptions

Field	Description
CHNERR	This register is used by the SDMA to warn the ARM platform when an incoming DMA request was detected and it triggers a channel that is already pending or being serviced. This probably means there is an overflow of data for that channel. • An interrupt is sent to the ARM platform if the corresponding channel bit is set in the INTRMASK register. • This is a "write-ones" register for the scheduler. It is only able to set the flags. The flags are cleared when the register is read by the ARM platform or during SDMA reset. • The CHNERR[i] bit is set when a DMA request that triggers channel *i* is received through the corresponding input pins and the EP[i] bit is already set; the EVTERR[i] bit is unaffected if the ARM platform tries to set the EP[i] bit, whereas, that EP[i] bit is already set.

55.8.11 Channel ARM platform Interrupt Mask (SDMAARM_INTRMASK)

Address: 20E_C000h base + 2Ch offset = 20E_C02Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	HIMASK																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_INTRMASK field descriptions

Field	Description
HIMASK	The Interrupt Mask Register contains 32 interrupt generation mask bits. If bit HIMASK[i] is set, the HI[i] bit is set and an interrupt is sent to the ARM platform when a DMA request error is detected on channel *i* (for example, EVTERR[i] is set).

55.8.12 Schedule Status (SDMAARM_PSW)

Address: 20E_C000h base + 30h offset = 20E_C030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																NCP[2:0]			NCR[4:0]					CCP[2:0]				CCR[4:0]			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_PSW field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15–13 NCP[2:0]	The Next Channel Priority gives the next pending channel priority. When the priority is 0, it means there is no pending channel and the NCR value has no meaning. 0 No running channel 1 Active channel priority
12–8 NCR[4:0]	The Next Channel Register indicates the number of the next scheduled pending channel with the highest priority.
7–4 CCP[2:0]	The Current Channel Priority indicates the priority of the current active channel. When the priority is 0, no channel is running: The SDMA is idle and the CCR value has no meaning. In the case that the SDMA has finished running the channel and has entered sleep state, CCP will indicate the priority of previous running channel. 0 No running channel 1 Active channel priority
CCR[4:0]	The Current Channel Register indicates the number of the channel that is being executed by the SDMA. SDMA. In the case that the SDMA has finished running the channel and has entered sleep state, CCR will indicate the previous running channel.

55.8.13 DMA Request Error Register (SDMAARM_EVTERRDBG)

Address: 20E_C000h base + 34h offset = 20E_C034h

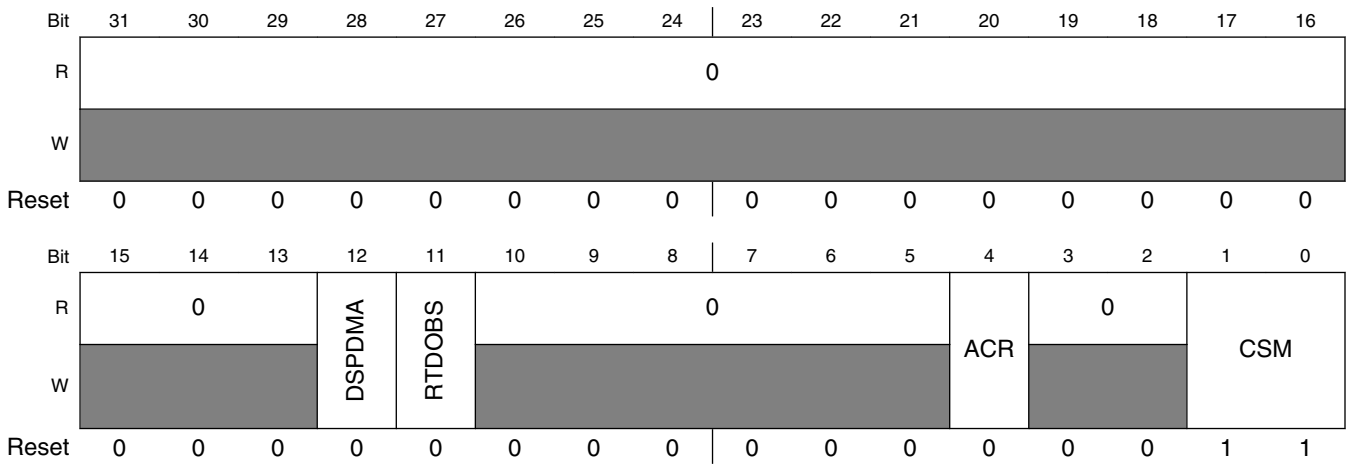
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CHNERR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_EVTERRDBG field descriptions

Field	Description
CHNERR	This register is the same as EVTERR, except reading it does not clear its contents. This address is meant to be used in debug mode. The ARM platform OnCE may check this register value without modifying it.

55.8.14 Configuration Register (SDMAARM_CONFIG)

Address: 20E_C000h base + 38h offset = 20E_C038h



SDMAARM_CONFIG field descriptions

Field	Description
31–13 Reserved	This read-only field is reserved and always has the value 0.
12 DSPDMA	This bit's function is reserved and should be configured as zero. 0 - Reset Value 1 - Reserved
11 RTDOBS	Indicates if Real-Time Debug pins are used: They do not toggle by default in order to reduce power consumption. 0 RTD pins disabled 1 RTD pins enabled
10–5 Reserved	This read-only field is reserved and always has the value 0.
4 ACR	ARM platform DMA / SDMA Core Clock Ratio. Selects the clock ratio between ARM platform DMA interfaces (burst DMA and peripheral DMA) and the internal SDMA core clock. The frequency selection is determined separately by the chip clock controller. This bit has to match the configuration of the chip clock controller that generates the clocks used in the SDMA. 0 ARM platform DMA interface frequency equals twice core frequency 1 ARM platform DMA interface frequency equals core frequency
3–2 Reserved	This read-only field is reserved and always has the value 0.

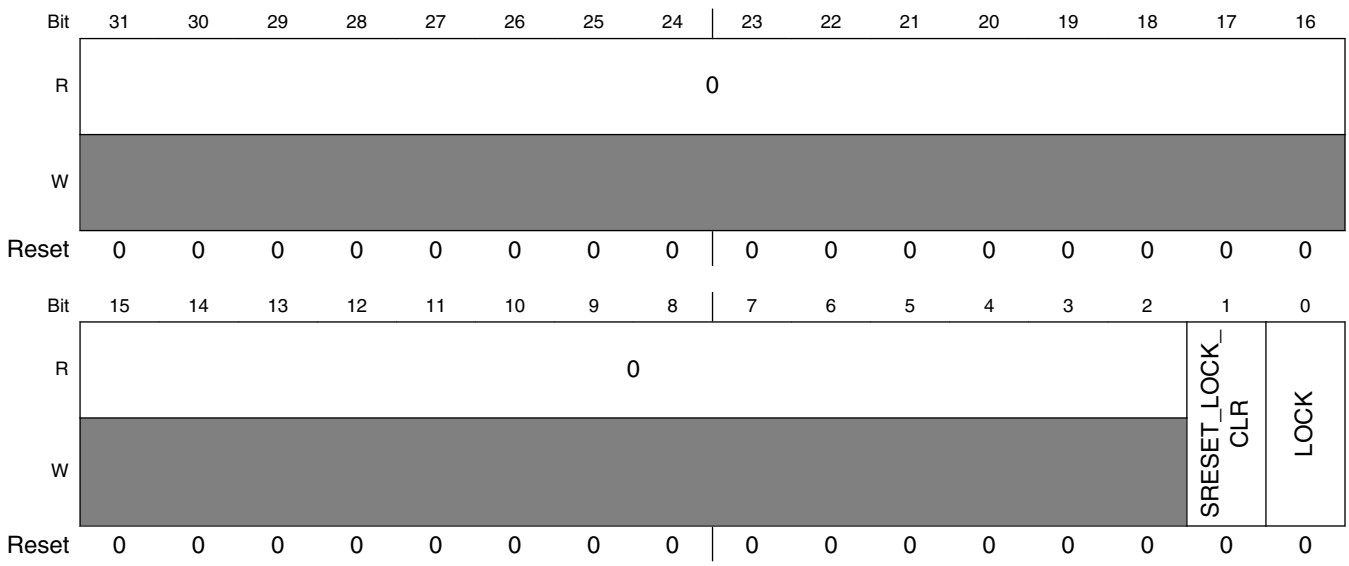
Table continues on the next page...

SDMAARM_CONFIG field descriptions (continued)

Field	Description
CSM	Selects the Context Switch Mode. The ARM platform has a read/write access. The SDMA cannot modify that register. The value at reset is 3, which selects the dynamic context switch by default. That register can be modified at anytime but the new context switch configuration will only be taken into account at the start of the next restore phase. NOTE: The first call to SDMA's channel 0 Bootload script after reset should use static context switch mode to ensure the context RAM for channel 0 is initialized in the channel SAVE Phase. After Channel 0 is run once, then any of the dynamic context modes can be used. 0 static 1 dynamic low power 2 dynamic with no loop 3 dynamic

55.8.15 SDMA LOCK (SDMAARM_SDMA_LOCK)

Address: 20E_C000h base + 3Ch offset = 20E_C03Ch



SDMAARM_SDMA_LOCK field descriptions

Field	Description
31–2 Reserved	This read-only field is reserved and always has the value 0.
1 SRESET_LOCK_CLR	The SRESET_LOCK_CLR bit determine if the LOCK bit is cleared on a software reset triggered by writing to the RESET register. This bit cannot be changed if LOCK=1. SREST_LOCK_CLR is cleared by conditions that clear the LOCK bit. 0 Software Reset does not clear the LOCK bit. 1 Software Reset clears the LOCK bit.

Table continues on the next page...

SDMAARM_SDMA_LOCK field descriptions (continued)

Field	Description
0 LOCK	The LOCK bit is used to restrict access to update SDMA script memory through ROM channel zero scripts and through the OnCE interface under ARM platform control. The LOCK bit is set: • The SDMA_LOCK, ONCE_ENB,CH0ADDR, and ILLINSTADDR registers cannot be written. These registers can be read, but writes are ignored. • SDMA software executing out of ROM or RAM may check the LOCK bit in the LOCK register Lock Status Register (SDMACORE_SDMA_LOCK) to determine if certain operations are allowed, such as up-loading new scripts. Once the LOCK bit is set to 1, only a reset can clear it. The LOCK bit is cleared by a hardware reset. LOCK is cleared by a software reset only if SRESET_LOCK_CLR is set. 0 LOCK disengaged. 1 LOCK enabled.

55.8.16 OnCE Enable (SDMAARM_ONCE_ENB)

Address: 20E_C000h base + 40h offset = 20E_C040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															ENB
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_ONCE_ENB field descriptions

Field	Description
31–1 Reserved	This read-only field is reserved and always has the value 0.
0 ENB	The OnCE Enable register selects the OnCE control source: When cleared (0), the OnCE registers are accessed through the JTAG interface; when set (1), the OnCE registers may be accessed by the ARM platform through the addresses described, as follows. • After reset, the OnCE registers are accessed through the JTAG interface. • Writing a 1 to ENB enables the ARM platform to access the ONCE_* as any other SDMA control register. • When cleared (0), all the ONCE_xxx registers cannot be written. The value of ENB cannot be changed if the LOCK bit in the SDMA_LOCK register is set.

55.8.17 OnCE Data Register (SDMAARM_ONCE_DATA)

Address: 20E_C000h base + 44h offset = 20E_C044h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_ONCE_DATA field descriptions

Field	Description
DATA	Data register of the OnCE JTAG controller. Refer to OnCE and Real-Time Debug for information on this register.

55.8.18 OnCE Instruction Register (SDMAARM_ONCE_INSTR)

Address: 20E_C000h base + 48h offset = 20E_C048h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																INSTR															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_ONCE_INSTR field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
INSTR	Instruction register of the OnCE JTAG controller. Refer to OnCE and Real-Time Debug for information on this register.

55.8.19 OnCE Status Register (SDMAARM_ONCE_STAT)

Address: 20E_C000h base + 4Ch offset = 20E_C04Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PST[3:0]				RCV	EDR	ODR	SWB	MST	0				ECDR		
W																
Reset	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_ONCE_STAT field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15–12 PST[3:0]	The Processor Status bits reflect the state of the SDMA RISC engine. Its states are as follows: • The "Program" state is the usual instruction execution cycle. • The "Data" state is inserted when there are wait-states during a load or a store on the data bus (ld or st). • The "Change of Flow" state is the second cycle of any instruction that breaks the sequence of instructions (jumps and channel switching instructions). • The "Change of Flow in Loop" state is used when an error causes a hardware loop exit. • The "Debug" state means the SDMA is in debug mode. • The "Functional Unit" state is inserted when there are wait-states during a load or a store on the functional units bus (ldf or stf). • In "Sleep" modes, no script is running (this is the RISC engine idle state). The "after Reset" is slightly different because no context restoring phase will happen when a channel is triggered: The script located at address 0 will be executed (boot operation). • The "in Sleep" states are the same as above except they do not have any corresponding channel: They are used when entering debug mode after reset. The reason is that it is necessary to return to the "Sleep after Reset" state when leaving debug mode. 0 Program 1 Data 2 Change of Flow 3 Change of Flow in Loop 4 Debug 5 Functional Unit 6 Sleep 7 Save 8 Program in Sleep 9 Data in Sleep 10 Change of Flow in Sleep 11 Change Flow in Loop in Sleep 12 Debug in Sleep 13 Functional Unit in Sleep 14 Sleep after Reset 15 Restore
11 RCV	After each write access to the real time buffer (RTB), the RCV bit is set. This bit is cleared after execution of an `rbuffer` command and on a JTAG reset.
10 EDR	This flag is raised when the SDMA has entered debug mode after an external debug request.
9 ODR	This flag is raised when the SDMA has entered debug mode after a OnCE debug request.
8 SWB	This flag is raised when the SDMA has entered debug mode after a software breakpoint.
7 MST	This flag is raised when the OnCE is controlled from the ARM platform peripheral interface. 0 The JTAG interface controls the OnCE. 1 The ARM platform peripheral interface controls the OnCE.
6–3 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

SDMAARM_ONCE_STAT field descriptions (continued)

Field	Description
ECDR	Event Cell Debug Request. If the debug request comes from the event cell, the reason for entering debug mode is given by the EDR bits. If all three bits of the EDR are reset, then it did not generate any debug request. If the cell did generate a debug request, then at least one of the EDR bits is set (the meaning of the encoding is given below). The encoding of the EDR bits is useful to find out more precisely why the debug request was generated. A debug request from an event cell is generated for a specific combination of the addra_cond, addrb_cond, and data_cond conditions. The value of those fields is given by the EDR bits. 0 1 matched addra_cond 1 1 matched addrb_cond 2 1 matched data_cond

55.8.20 OnCE Command Register (SDMAARM_ONCE_CMD)

Address: 20E_C000h base + 50h offset = 20E_C050h

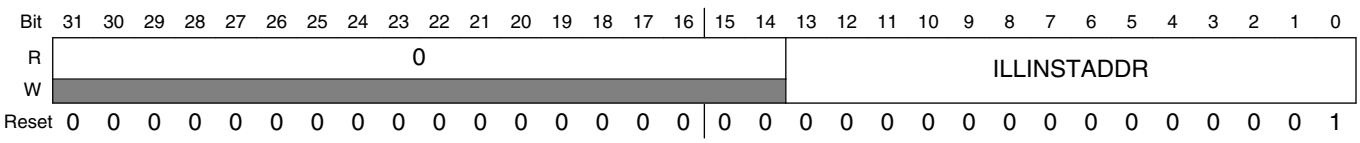
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																												CMD			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_ONCE_CMD field descriptions

Field	Description
31–4 Reserved	This read-only field is reserved and always has the value 0.
CMD	Writing to this register will cause the OnCE to execute the command that is written. When needed, the ONCE_DATA and ONCE_INSTR registers should be loaded with the correct value before writing the command to that register. For a list of the OnCE commands and their usage, see OnCE and Real-Time Debug. **NOTE:** 7-15 *reserved* 0 rstatus 1 dmov 2 exec_once 3 run_core 4 exec_core 5 debug_rqst 6 rbuffer

55.8.21 Illegal Instruction Trap Address (SDMAARM_ILLINSTADDR)

Address: 20E_C000h base + 58h offset = 20E_C058h

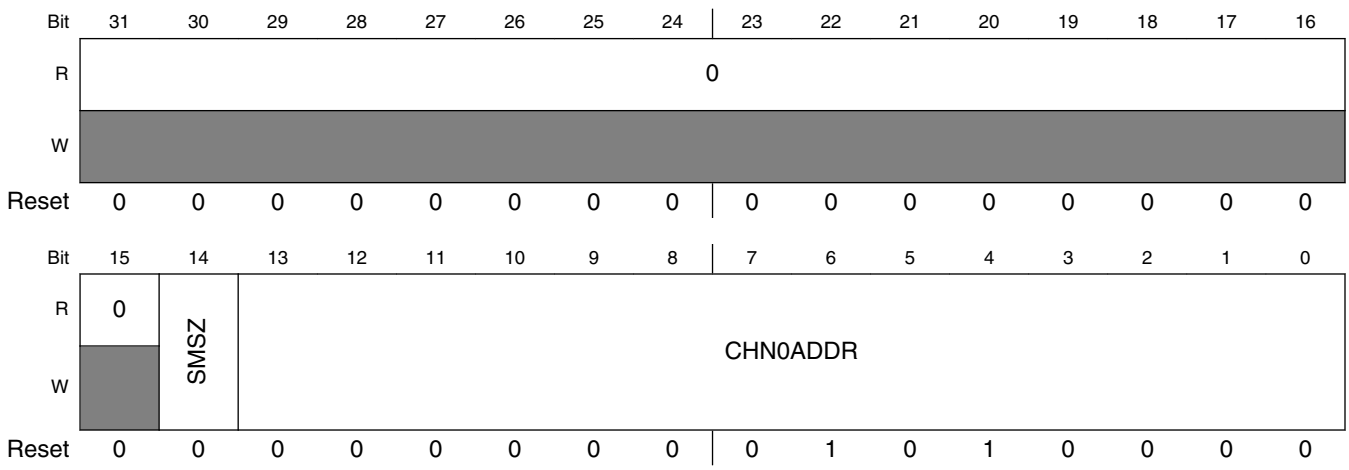


SDMAARM_ILLINSTADDR field descriptions

Field	Description
31–14 Reserved	This read-only field is reserved and always has the value 0.
ILLINSTADDR	The Illegal Instruction Trap Address is the address where the SDMA jumps when an illegal instruction is executed. It is 0x0001 after reset. The value of ILLINSTADDR cannot be changed if the LOCK bit in the SDMA_LOCK register is set.

55.8.22 Channel 0 Boot Address (SDMAARM_CHN0ADDR)

Address: 20E_C000h base + 5Ch offset = 20E_C05Ch



SDMAARM_CHN0ADDR field descriptions

Field	Description
31–15 Reserved	This read-only field is reserved and always has the value 0.
14 SMSZ	The bit 14 (Scratch Memory Size) determines if scratch memory must be available after every channel context. After reset, it is equal to 0, which defines a RAM space of 24 words for each channel. All of this area stores the channel context. By setting this bit, 32 words are reserved for every channel context,

Table continues on the next page...

SDMAARM_CHN0ADDR field descriptions (continued)

Field	Description
	which gives eight additional words that can be used by the channel script to store any type of data. Those words are never erased by the context switching mechanism. The value of SMSZ cannot be changed if the LOCK bit in the SDMA_LOCK register is set. 0 24 words per context 1 32 words per context
CHN0ADDR	This 14-bit register is used by the boot code of the SDMA. After reset, it points to the standard boot routine in ROM (channel 0 routine). By changing this address, you can perform a boot sequence with your own routine. The very first instructions of the boot code fetch the contents of this register (it is also mapped in the SDMA memory space) and jump to the given address. The reset value is 0x0050 (decimal 80). The value of CHN0ADDR cannot be changed if the LOCK bit in the SDMA_LOCK register is set.

55.8.23 DMA Requests (SDMAARM_EVT_MIRROR)

Address: 20E_C000h base + 60h offset = 20E_C060h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EVENTS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_EVT_MIRROR field descriptions

Field	Description
EVENTS	This register reflects the DMA requests received by the SDMA for events 31-0. The ARM platform and the SDMA have a read-only access. There is one bit associated with each of 32 DMA request events. This information may be useful during debug of the blocks that generate the DMA requests. The EVT_MIRROR register is cleared following read access. 0 DMA request event not pending 1 DMA request event pending

55.8.24 DMA Requests 2 (SDMAARM_EVT_MIRROR2)

Address: 20E_C000h base + 64h offset = 20E_C064h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																EVENTS[47:32]															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_EVT_MIRROR2 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
EVENTS[47:32]	This register reflects the DMA requests received by the SDMA for events 47-32. The ARM platform and the SDMA have a read-only access. There is one bit associated with each of DMA request events. This information may be useful during debug of the blocks that generate the DMA requests. The EVT_MIRROR2 register is cleared following read access. 0 - DMA request event not pending 1- DMA request event pending

55.8.25 Cross-Trigger Events Configuration Register 1 (SDMAARM_XTRIG_CONF1)

Address: 20E_C000h base + 70h offset = 20E_C070h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	CNF3	NUM3[5:0]						0	CNF2	NUM2[5:0]					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	CNF1	NUM1[5:0]						0	CNF0	NUM0[5:0]					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_XTRIG_CONF1 field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30 CNF3	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by the reception of a DMA request or by the starting of a channel script execution. 0 channel 1 DMA request
29–24 NUM3[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.
23 Reserved	This read-only field is reserved and always has the value 0.

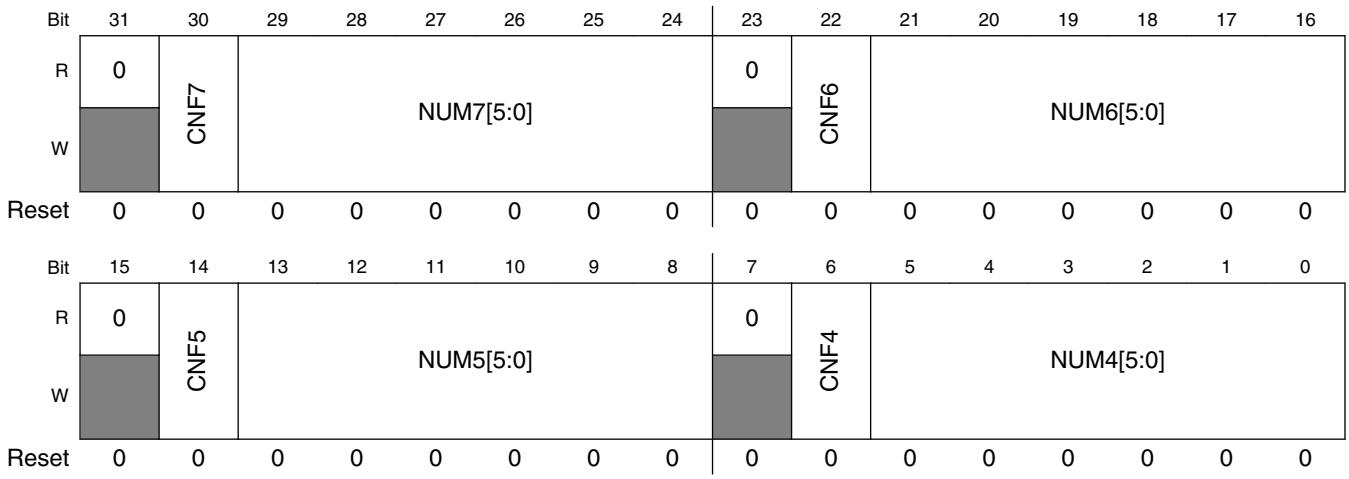
Table continues on the next page...

SDMAARM_XTRIG_CONF1 field descriptions (continued)

Field	Description
22 CNF2	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution. 0 channel 1 DMA request
21–16 NUM2[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.
15 Reserved	This read-only field is reserved and always has the value 0.
14 CNF1	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution. 0 channel 1 DMA request
13–8 NUM1[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.
7 Reserved	This read-only field is reserved and always has the value 0.
6 CNF0	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution. 0 channel 1 DMA request
NUM0[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.

55.8.26 Cross-Trigger Events Configuration Register 2 (SDMAARM_XTRIG_CONF2)

Address: 20E_C000h base + 74h offset = 20E_C074h



SDMAARM_XTRIG_CONF2 field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30 CNF7	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution. 0 channel 1 DMA request
29–24 NUM7[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.
23 Reserved	This read-only field is reserved and always has the value 0.
22 CNF6	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution. 0 channel 1 DMA request
21–16 NUM6[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.
15 Reserved	This read-only field is reserved and always has the value 0.
14 CNF5	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution

Table continues on the next page...

SDMAARM_XTRIG_CONF2 field descriptions (continued)

Field	Description
	0 channel 1 DMA request
13–8 NUM5[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.
7 Reserved	This read-only field is reserved and always has the value 0.
6 CNF4	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution. 0 channel 1 DMA request
NUM4[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.

55.8.27 Channel Priority Registers (SDMAARM_SDMA_CHNPRI*n*)

Address: 20E_C000h base + 100h offset + (4d × i), where i=0d to 31d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																													CHNPRIn		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_SDMA_CHNPRI*n* field descriptions

Field	Description
31–3 Reserved	This read-only field is reserved and always has the value 0.
CHNPRIn	This contains the priority of channel number *n*. Useful values are between 1 and 7; 0 is reserved by the SDMA hardware to determine when there is no pending channel. Reset value is 0, which prevents the channels from starting.

55.8.28 Channel Enable RAM (SDMAARM_CHNENBL*n*)

Address: 20E_C000h base + 200h offset + (4d × i), where i=0d to 47d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ENBLn																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_CHNENBL*n* field descriptions

Field	Description
ENBLn	This 32-bit value selects the channels that are triggered by the DMA request number *n*. If ENBLn[i] is set to 1, bit EP[i] will be set when the DMA request *n* is received. These 48 32-bit registers are physically located in a RAM, with no known reset value. It is thus essential for the ARM platform to program them before any DMA request is triggered to the SDMA, otherwise an unpredictable combination of channels may be started.

55.9 BP Memory Map and Control Register Definitions

The following section describes SDMA control registers available to the BP.

NOTE

These registers are physically implemented in all platforms, but are not accessible when the SDMA BP control port is not connected. Reset values are calculated to allow the system to work when those registers cannot be accessed.

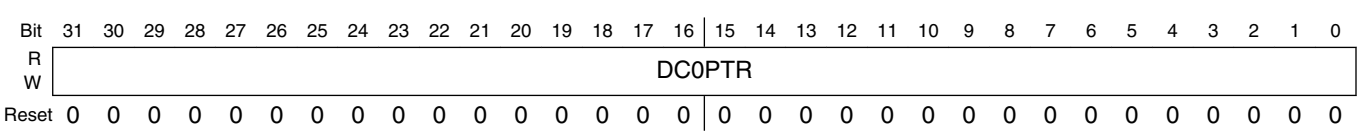
All registers are clocked with the SDMA clock (which means the SDMA clock must be running when the BP wants to access any register).

SDMABP memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20E_C000	Channel 0 Pointer (SDMABP_DC0PTR)	32	R/W	0000_0000h	55.9.1/4912
20E_C004	Channel Interrupts (SDMABP_INTR)	32	w1c	0000_0000h	55.9.2/4913
20E_C008	Channel Stop/Channel Status (SDMABP_STOP_STAT)	32	R/W	0000_0000h	55.9.3/4913
20E_C00C	Channel Start (SDMABP_DSTART)	32	R	0000_0000h	55.9.4/4914
20E_C028	DMA Request Error Register (SDMABP_EVTERR)	32	R	0000_0000h	55.9.5/4914
20E_C02C	Channel DSP Interrupt Mask (SDMABP_INTRMASK)	32	R/W	0000_0000h	55.9.6/4915
20E_C034	DMA Request Error Register (SDMABP_EVTERRDBG)	32	R	0000_0000h	55.9.7/4915

55.9.1 Channel 0 Pointer (SDMABP_DC0PTR)

Address: 20E_C000h base + 0h offset = 20E_C000h



SDMABP_DC0PTR field descriptions

Field	Description
DC0PTR	**Channel 0 Pointer** contains the 32-bit address, in BP memory, of the array of channel control blocks starting with the one for channel 0 (the control channel). This register should be initialized by the BP before it enables a channel (for example, channel 0). See the API document SDMA Scripts User Manual for the use of this register. The BP has a read/write access and the SDMA has a read-only access.

55.9.2 Channel Interrupts (SDMABP_INTR)

Address: 20E_C000h base + 4h offset = 20E_C004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DI																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMABP_INTR field descriptions

Field	Description
DI	The BP Interrupts register contains the 32 DI[i] bits. If any bit is set, it will cause an interrupt to the BP. • This register is a "write-ones" register to the BP. When the BP sets a bit in this register, the corresponding DI[i] bit is cleared. • The interrupt service routine should clear individual channel bits when their interrupts are serviced; failure to do so will cause continuous interrupts. • The SDMA is responsible for setting the DI[i] bit corresponding to the current channel when the corresponding `done` instruction is executed.

55.9.3 Channel Stop/Channel Status (SDMABP_STOP_STAT)

Address: 20E_C000h base + 8h offset = 20E_C008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DE																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMABP_STOP_STAT field descriptions

Field	Description
DE	This 32-bit register gives access to the BP (DSP) Enable bits, DE. There is one bit for every channel. • This register is a "write-ones" register to the BP. • When the BP writes 1 in bit *i* of this register, it clears the DE[i] and DSTART[i] bits. • Reading this register yields the current state of the DE[i] bits.

55.9.4 Channel Start (SDMABP_DSTART)

Address: 20E_C000h base + Ch offset = 20E_C00Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DSTART_DE																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMABP_DSTART field descriptions

Field	Description
DSTART_DE	The DSTART_DE registers are 32 bits wide with one bit for every channel. • When a bit is written to 1, it enables the corresponding channel. • Two physical registers are accessed with that address (DSTART and DE), which enables the BP to trigger a channel a second time before the first trigger was processed. • This register is a "write-ones" register to the BP. Neither DSTART[i] bit can be set while the corresponding DE[i] bit is cleared. • When the BP tries to set the DSTART[i] bit by writing a one (if the corresponding DE[i] bit is clear), the bit in the DSTART[i] register will remain cleared and the DE[i] bit will be set. If the corresponding DE[i] bit was already set, the DSTART[i] bit will be set. • The next time the SDMA channel *i* attempts to clear the DE[i] bit by means of a `done` instruction, the bit in the DSTART[i] register will be cleared and the DE[i] bit will take the old value of the DSTART[i] bit. • Reading this register yields the current state of the DSTART[i] bits. This mechanism enables the BP to pipeline two DSTART commands per channel.

55.9.5 DMA Request Error Register (SDMABP_EVTERR)

Address: 20E_C000h base + 28h offset = 20E_C028h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CHNERR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMABP_EVTERR field descriptions

Field	Description
CHNERR	This register is used by the SDMA to warn the BP when an incoming DMA request was detected; it then triggers a channel that is already pending or being serviced, which may mean there is an overflow of data for that channel. An interrupt is sent to the BP if the corresponding channel bit is set in the INTRMASK register. • This is a "write-ones" register for the scheduler. It is only able to set the flags. The flags are cleared when the register is read by the BP or during an SDMA reset. • The CHNERR[i] bit is set when a DMA request that triggers channel *i* is received through the corresponding input pins and the EP[i] bit is already set. The EVTERR[i] bit is unaffected if the BP tries to set the EP[i] bit when that EP[i] bit is already set.

55.9.6 Channel DSP Interrupt Mask (SDMABP_INTRMASK)

Address: 20E_C000h base + 2Ch offset = 20E_C02Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DIMASK																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMABP_INTRMASK field descriptions

Field	Description
DIMASK	The Interrupt Mask Register contains 32 interrupt generation mask bits. If bit DIMASK[i] is set, the DI[i] bit is set and an interrupt is sent to the BP when a DMA request error is detected on channel *i* (for example, EVTERR[i] is set).

55.9.7 DMA Request Error Register (SDMABP_EVTERRDBG)

Address: 20E_C000h base + 34h offset = 20E_C034h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CHNERR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMABP_EVTERRDBG field descriptions

Field	Description
CHNERR	This register is the same as EVTERR except reading it does not clear its contents. This address is meant to be used in debug mode. The BP OnCE may check this register value without modifying it.

55.10 SDMA Internal (Core) Memory Map and Internal Register Definitions

The actual SDMA memory mapped registers are summarized in the following sections; for peripherals' memory maps, refer to the respective chapters.

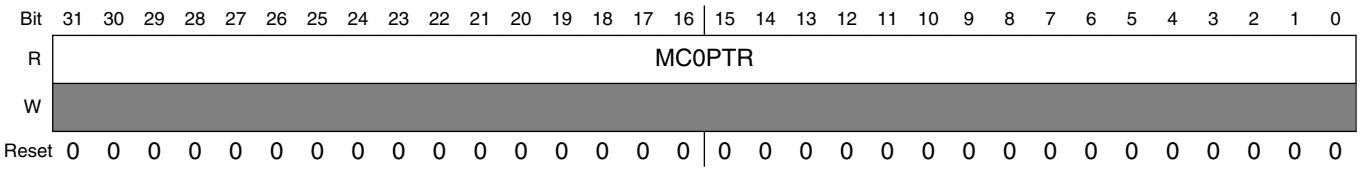
The following definitions serve as a key for the SDMA internal register summary.

SDMACORE memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20E_C000	ARM platform Channel 0 Pointer (SDMACORE_MC0PTR)	32	R	0000_0000h	55.10.1/ 4917
20E_C002	Current Channel Pointer (SDMACORE_CCPTR)	32	R	0000_0000h	55.10.2/ 4917
20E_C003	Current Channel Register (SDMACORE_CCR)	32	R	0000_0000h	55.10.3/ 4917
20E_C004	Highest Pending Channel Register (SDMACORE_NCR)	32	R	0000_0000h	55.10.4/ 4918
20E_C005	External DMA Requests Mirror (SDMACORE_EVENTS)	32	R	0000_0000h	55.10.5/ 4919
20E_C006	Current Channel Priority (SDMACORE_CCPRI)	32	R	0000_0000h	55.10.6/ 4920
20E_C007	Next Channel Priority (SDMACORE_NCPRI)	32	R	0000_0000h	55.10.7/ 4920
20E_C009	OnCE Event Cell Counter (SDMACORE_ECOUNT)	32	R/W	0000_0000h	55.10.8/ 4921
20E_C00A	OnCE Event Cell Control Register (SDMACORE_ECTL)	32	R/W	0000_0000h	55.10.9/ 4921
20E_C00B	OnCE Event Address Register A (SDMACORE_EAA)	32	R/W	0000_0000h	55.10.10/ 4923
20E_C00C	OnCE Event Cell Address Register B (SDMACORE_EAB)	32	R/W	0000_0000h	55.10.11/ 4923
20E_C00D	OnCE Event Cell Address Mask (SDMACORE_EAM)	32	R/W	0000_0000h	55.10.12/ 4923
20E_C00E	OnCE Event Cell Data Register (SDMACORE_ED)	32	R/W	0000_0000h	55.10.13/ 4924
20E_C00F	OnCE Event Cell Data Mask (SDMACORE_EDM)	32	R/W	0000_0000h	55.10.14/ 4924
20E_C018	OnCE Real-Time Buffer (SDMACORE_RTB)	32	R/W	0000_0000h	55.10.15/ 4925
20E_C019	OnCE Trace Buffer (SDMACORE_TB)	32	R	0000_0000h	55.10.16/ 4925
20E_C01A	OnCE Status (SDMACORE_OSTAT)	32	R	0000_0000h	55.10.17/ 4926
20E_C01C	Channel 0 Boot Address (SDMACORE_MCHN0ADDR)	32	R	0000_0000h	55.10.18/ 4928
20E_C01D	ENDIAN Status Register (SDMACORE_ENDIANNESS)	32	R	0000_0001h	55.10.19/ 4929
20E_C01E	Lock Status Register (SDMACORE_SDMA_LOCK)	32	R	0000_0000h	55.10.20/ 4930
20E_C01F	External DMA Requests Mirror #2 (SDMACORE_EVENTS2)	32	R	0000_0000h	55.10.21/ 4931

55.10.1 ARM platform Channel 0 Pointer (SDMACORE_MC0PTR)

Address: 20E_C000h base + 0h offset = 20E_C000h

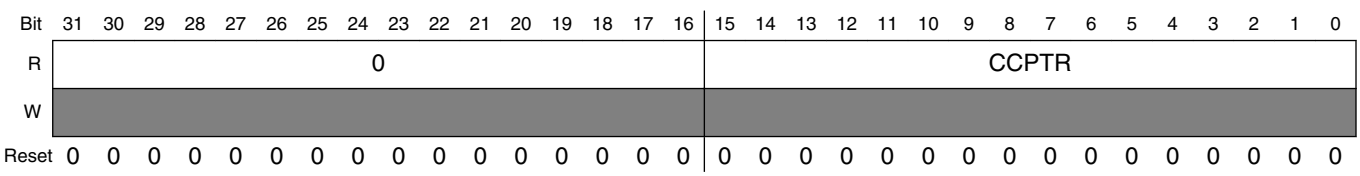


SDMACORE_MC0PTR field descriptions

Field	Description
MC0PTR	Contains the address-in the ARM platform memory space-of the initial SDMA context and scripts that are loaded by the SDMA boot script running on channel 0.

55.10.2 Current Channel Pointer (SDMACORE_CCPTR)

Address: 20E_C000h base + 2h offset = 20E_C002h

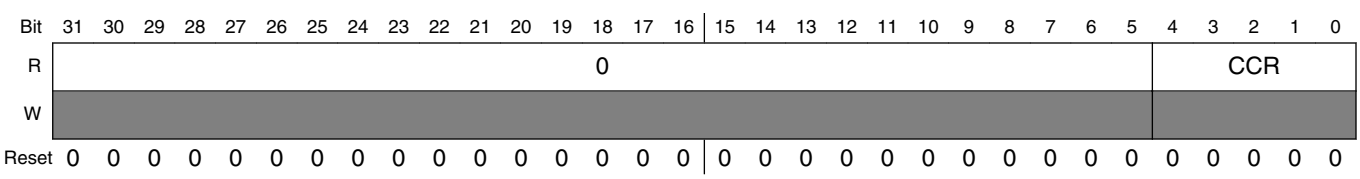


SDMACORE_CCPTR field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
CCPTR	Contains the start address of the context data for the current channel: Its value is *CONTEXT_BASE* + 24* *CCR* or *CONTEXT_BASE* + 32* *CCR* where CONTEXT_BASE = 0x0800. The value 24 or 32 is selected according to the programmed channel scratch RAM size in the register shown in Channel 0 Boot Address (SDMAARM_CHN0ADDR) .

55.10.3 Current Channel Register (SDMACORE_CCR)

Address: 20E_C000h base + 3h offset = 20E_C003h



SDMACORE_CCR field descriptions

Field	Description
31–5 Reserved	This read-only field is reserved and always has the value 0.
CCR	Contains the number of the current running channel whose context is installed. In the case that the SDMA has finished running the channel and has entered sleep state, CCR will indicate the previous running channel. The PST bits in the OSTAT register indicate when the SDMA is in sleep state.

55.10.4 Highest Pending Channel Register (SDMACORE_NCR)

Address: 20E_C000h base + 4h offset = 20E_C004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																											NCR				
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMACORE_NCR field descriptions

Field	Description
31–5 Reserved	This read-only field is reserved and always has the value 0.
NCR	Contains the number of the pending channel that the scheduler has selected to run next.

55.10.5 External DMA Requests Mirror (SDMACORE_EVENTS)

NOTE

This register is very useful in the case of DMA requests that are active when a peripheral FIFO level is above the programmed watermark. The activation of the DMA request (rising edge) is detected by the SDMA logic and it can enable one or several channels. One of the channels accesses the peripheral and reads or writes a number of data that matches the watermark level (for example, if the watermark is four words, the channel reads or writes four words).

If the channel is effectively executed long after the DMA request was received, reading or writing the watermark number of data may not be sufficient to reset the DMA request (for example, if the FIFO watermark is four and at the channel execution it already contains nine pieces of data). This means no new rising edge may be detected by the SDMA, although there still remains transfers to perform. Therefore, if the channel were terminated at that time, it would not be restarted, causing potential overrun or underrun of the peripheral.

The proposed mechanism is for the channel to check this register after it has performed the "watermark" number of accesses to the peripheral. If the bit for the DMA request that triggers this channel is set, it means there is still another watermark number of data to transfer. This goes on until the bit is cleared. The same script can be used for multiple channels that require this behavior. The script can determine its channel number from the CCR register and infer the corresponding DMA request bit to check. It needs a reference table that is coherent with the request-channel matrix that the ARM platform programmed.

Address: 20E_C000h base + 5h offset = 20E_C005h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EVENTS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMACORE_EVENTS field descriptions

Field	Description
EVENTS	Reflects the status of the SDMA's external DMA requests. It is meant to allow any channel to monitor the states of these SDMA inputs. This register displays EVENTS 0-31. The EVENTS2 register displays events 32-47.

55.10.6 Current Channel Priority (SDMACORE_CCPRI)

Address: 20E_C000h base + 6h offset = 20E_C006h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																													CCPRI		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMACORE_CCPRI field descriptions

Field	Description
31–3 Reserved	This read-only field is reserved and always has the value 0.
CCPRI	Contains the 3-bit priority of the channel whose context is installed. It is 0 when no channel is running. **NOTE:** 1-7 current channel priority 0 no running channel

55.10.7 Next Channel Priority (SDMACORE_NCPRI)

Address: 20E_C000h base + 7h offset = 20E_C007h

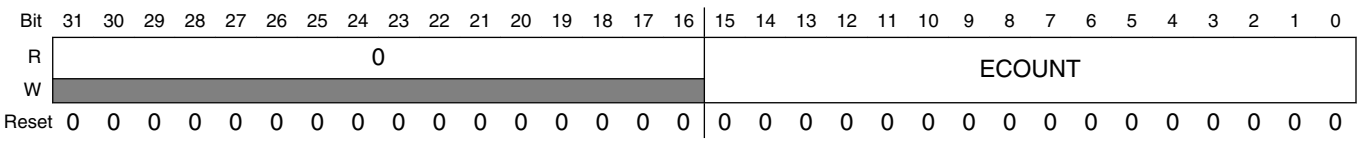
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																													NCPRI		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMACORE_NCPRI field descriptions

Field	Description
31–3 Reserved	This read-only field is reserved and always has the value 0.
NCPRI	Contains the 3-bit priority of the channel the scheduler has selected to run next. It is 0 when no other channel is pending.

55.10.8 OnCE Event Cell Counter (SDMACORE_ECOUNT)

Address: 20E_C000h base + 9h offset = 20E_C009h

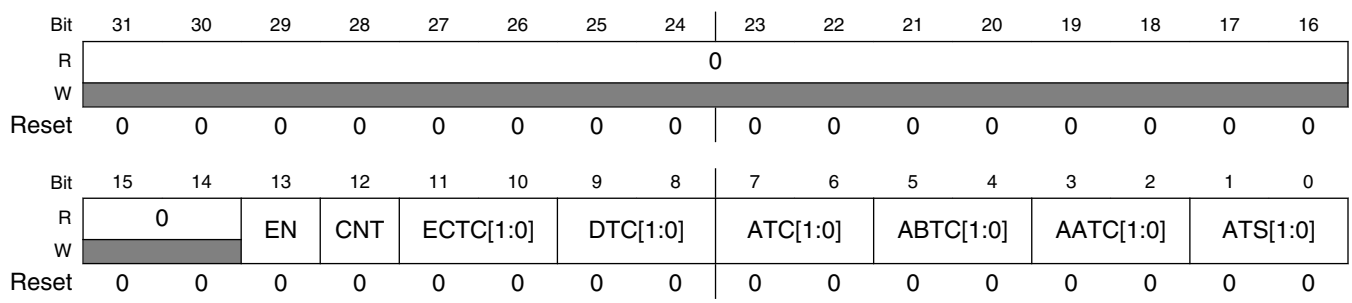


SDMACORE_ECOUNT field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
ECOUNT	The event cell counter contains the number of times minus one that an event detection must occur before generating a debug request. • This register should be written before any attempt to use the event detection counter during an event detection process. • The counter is cleared on a JTAG reset.

55.10.9 OnCE Event Cell Control Register (SDMACORE_ECTL)

Address: 20E_C000h base + Ah offset = 20E_C00Ah



SDMACORE_ECTL field descriptions

Field	Description
31–14 Reserved	This read-only field is reserved and always has the value 0.
13 EN	Event Cell Enable. If the EN bit is set, the event cell is allowed to generate debug requests (the cell is awakened). If it is cleared, the event detection unit is disabled and no hardware breakpoint is generated, but matching conditions are still reflected on the emulation pin. 0 Cell is disabled. 1 Cell is enabled.
12 CNT	Event Counter Enable. The event counter enable bit determines if the cell counter is used during the event detection. In order to use the event counter during an event detection process, the event cell counter register should be loaded with a value equal to the number of times minus one that an event occurs before

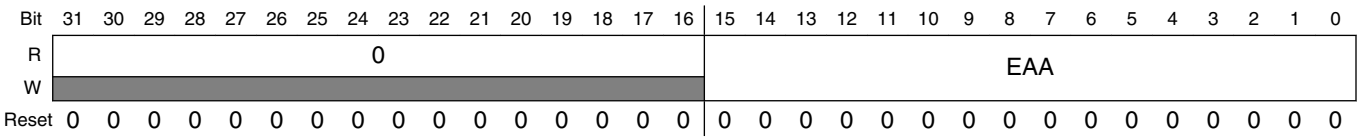
Table continues on the next page...

SDMACORE_ECTL field descriptions (continued)

Field	Description
	a debug request is sent. After every event detection, the counter is decreased. When the counter reaches the value 0, the event detection cell sends a debug request to the core. The event counter register should be written and the EN bit should be set before each new event detection process uses the event counter. 0 Counter is disabled. 1 Counter is enabled.
11–10 ECTC[1:0]	The event cell trigger condition bits select the combination of address and data matching conditions that generate the final address/data condition. During program execution, if this event cell trigger condition goes to 1, a debug request is sent to the SDMA. The EN bit must be set to enable the debug request generation. 00 address ONLY 01 data ONLY 10 address AND data 11 address OR data
9–8 DTC[1:0]	The data trigger condition bits define when data is considered matching after comparison with the data register of the event detection unit. The operations are performed on unsigned values. 00 equal 01 not equal 10 greater than 11 less than
7–6 ATC[1:0]	The address trigger condition bits select how the two address conditions (addressA and addressB) are combined to define the global address matching condition. The supported combinations are described, as follows. 00 addressA ONLY 01 addrA AND addrB 10 addrA OR addrB 11 reserved
5–4 ABTC[1:0]	The Address B Trigger Condition (ABTC) controls the operations performed by address comparator B. All operations are performed on unsigned values. This comparator B outputs the addressB condition. 00 equal 01 not equal 10 greater than 11 less than
3–2 AATC[1:0]	The Address A Trigger Condition (AATC) controls the operations performed by address comparator A. All operations are performed on unsigned values. This comparator A outputs the addressA condition. 00 equal 01 not equal 10 greater than 11 less than
ATS[1:0]	The access type select bits define the memory access type required on the SDMA memory bus. 00 read ONLY 01 write ONLY 10 read or write 11 -

55.10.10 OnCE Event Address Register A (SDMACORE_EAA)

Address: 20E_C000h base + Bh offset = 20E_C00Bh

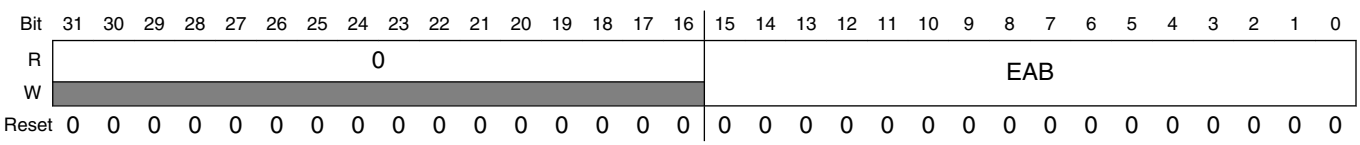


SDMACORE_EAA field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
EAA	Event Cell Address Register A computes an address A condition. It is cleared on a JTAG reset.

55.10.11 OnCE Event Cell Address Register B (SDMACORE_EAB)

Address: 20E_C000h base + Ch offset = 20E_C00Ch

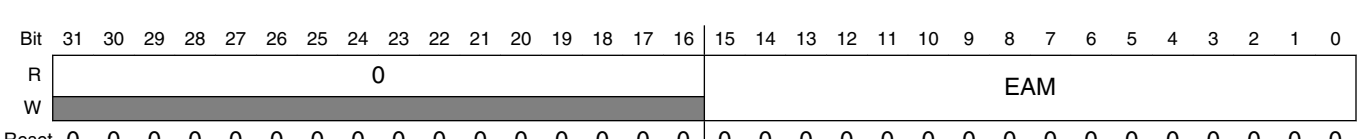


SDMACORE_EAB field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
EAB	Event Cell Address Register B computes an address B condition. It is cleared on a JTAG reset.

55.10.12 OnCE Event Cell Address Mask (SDMACORE_EAM)

Address: 20E_C000h base + Dh offset = 20E_C00Dh



SDMACORE_EAM field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
EAM	The Event Cell Address Mask contains a user-defined address mask value. This mask is applied to the address value latched from the memory address bus before performing the address comparison. **NOTE:** There is a common address mask value for both address comparators. If bit *i* of this register is set, then bit *i* of the address value latched from the memory bus does not influence the result of the address comparison. The register is cleared on a JTAG reset.

55.10.13 OnCE Event Cell Data Register (SDMACORE_ED)

Address: 20E_C000h base + Eh offset = 20E_C00Eh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ED																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMACORE_ED field descriptions

Field	Description
ED	The event cell data register contains a user defined data value. This data value is an input for the data comparator which generates the data condition. It is cleared on a JTAG reset.

55.10.14 OnCE Event Cell Data Mask (SDMACORE_EDM)

Address: 20E_C000h base + Fh offset = 20E_C00Fh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	EDM																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMACORE_EDM field descriptions

Field	Description
EDM	The event cell data mask register contains the user-defined data mask value. • This mask is applied to the data value latched from the memory bus before performing the data comparison. • Setting bit *i* of the event cell data mask register means that bit *i* of the data value latched from the address bus does not influence the result of the data comparison. • The data mask is cleared on a JTAG reset.

55.10.15 OnCE Real-Time Buffer (SDMACORE_RTB)

Address: 20E_C000h base + 18h offset = 20E_C018h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RTB																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMACORE_RTB field descriptions

Field	Description
RTB	The Real Time Buffer register stores and retrieves run time information without putting the SDMA in debug mode. Writing to that register triggers a pulse on a specific real-time debug pin whose connection depends on the chip implementation. The RTB value can be accessed by the OnCE under ARM platform or JTAG control using the rbuffer command.

55.10.16 OnCE Trace Buffer (SDMACORE_TB)

Address: 20E_C000h base + 19h offset = 20E_C019h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			TBF	TADDR											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TADDR		CHFADDR													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMACORE_TB field descriptions

Field	Description
31–29 Reserved	This read-only field is reserved and always has the value 0.
28 TBF	The Trace Buffer Flag is set when the buffer contains the addresses of a valid change of flow. The contents of the buffer should be ignored otherwise. 0 Invalid information 1 Valid information
27–14 TADDR	The target address is the address taken after the execution of the change of flow instruction.
CHFADDR	The change of flow address is the address where the change of flow is taken when executing a change of flow instruction.

55.10.17 OnCE Status (SDMACORE_OSTAT)

Address: 20E_C000h base + 1Ah offset = 20E_C01Ah

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PST[3:0]				RCV	EDR	ODR	SWB	MST	0				ECDR[2:0]		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMACORE_OSTAT field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15–12 PST[3:0]	The Processor Status bits reflect the state of the SDMA RISC engine. • The "Program" state is the usual instruction execution cycle. • The "Data" state is inserted when there are wait-states during a load or a store on the data bus (ld or st). • The "Change of Flow" state is the second cycle of any instruction that breaks the sequence of instructions (jumps and channel-switching instructions). • The "Change of Flow in Loop" state is used when an error causes a hardware loop exit. • The "Debug" state means the SDMA is in debug mode. • The "Functional Unit" state is inserted when there are wait-states during a load or a store on the functional units bus (ldf or stf). • In "Sleep" modes, no script is running (this is the RISC engine idle state). The "after Reset" is slightly different because no context restoring phase will happen when a channel is triggered: The script located at address 0 will be executed (boot operation). • The "in Sleep" states are the same as above except they do not have any corresponding channel. They are used when entering debug mode after reset; the reason is that it is necessary to return to the "Sleep after Reset" state when leaving debug mode. 0 Program 1 Data 2 Change of Flow 3 Change of Flow in Loop 4 Debug 5 Functional Unit 6 Sleep 7 Save 8 Program in Sleep 9 Data in Sleep 10 Change of Flow in Sleep 11 Change Flow Loop Sleep 12 Debug in Sleep 13 Functional Unit in Sleep

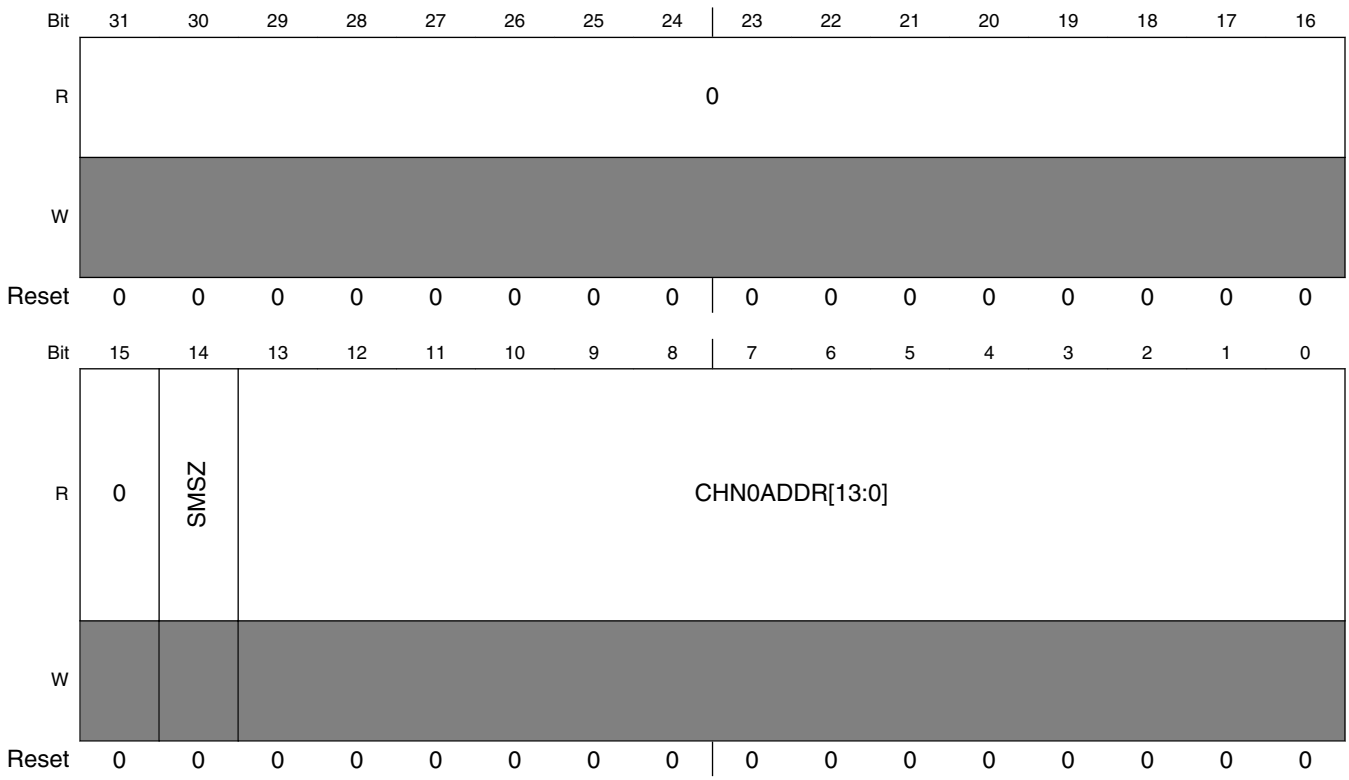
Table continues on the next page...

SDMACORE_OSTAT field descriptions (continued)

Field	Description
	14 Sleep after Reset 15 Restore
11 RCV	After each write access to the real time buffer (RTB), the RCV bit is set. This bit is cleared after execution of an `rbuffer` command and on a JTAG reset.
10 EDR	This flag is raised when the SDMA has entered debug mode after an external debug request.
9 ODR	This flag is raised when the SDMA has entered debug mode after a OnCE debug request.
8 SWB	This flag is raised when the SDMA has entered debug mode after a software breakpoint.
7 MST	This flag is raised when the OnCE is controlled from the ARM platform peripheral interface. 0 JTAG interface controls the OnCE. 1 ARM platform peripheral interface controls the OnCE.
6–3 Reserved	This read-only field is reserved and always has the value 0.
ECDR[2:0]	Event Cell Debug Request. If the debug request comes from the event cell, the reason for entering debug mode is given by the EDR bits. The encoding of the EDR bits is useful to find out more precisely why the debug request was generated. A debug request from an event cell is generated for a specific combination of the addressA, addressB, and data conditions; the value of those fields is given by the EDR bits. If all three bits of the EDR are reset, then it did not generate any debug request. If the cell did generate a debug request, then at least one EDR bit is set; the meaning of the encoding is as follows: 0 1 matched addressA condition 1 1 matched addressB condition 2 1 matched data condition

55.10.18 Channel 0 Boot Address (SDMACORE_MCHN0ADDR)

Address: 20E_C000h base + 1Ch offset = 20E_C01Ch

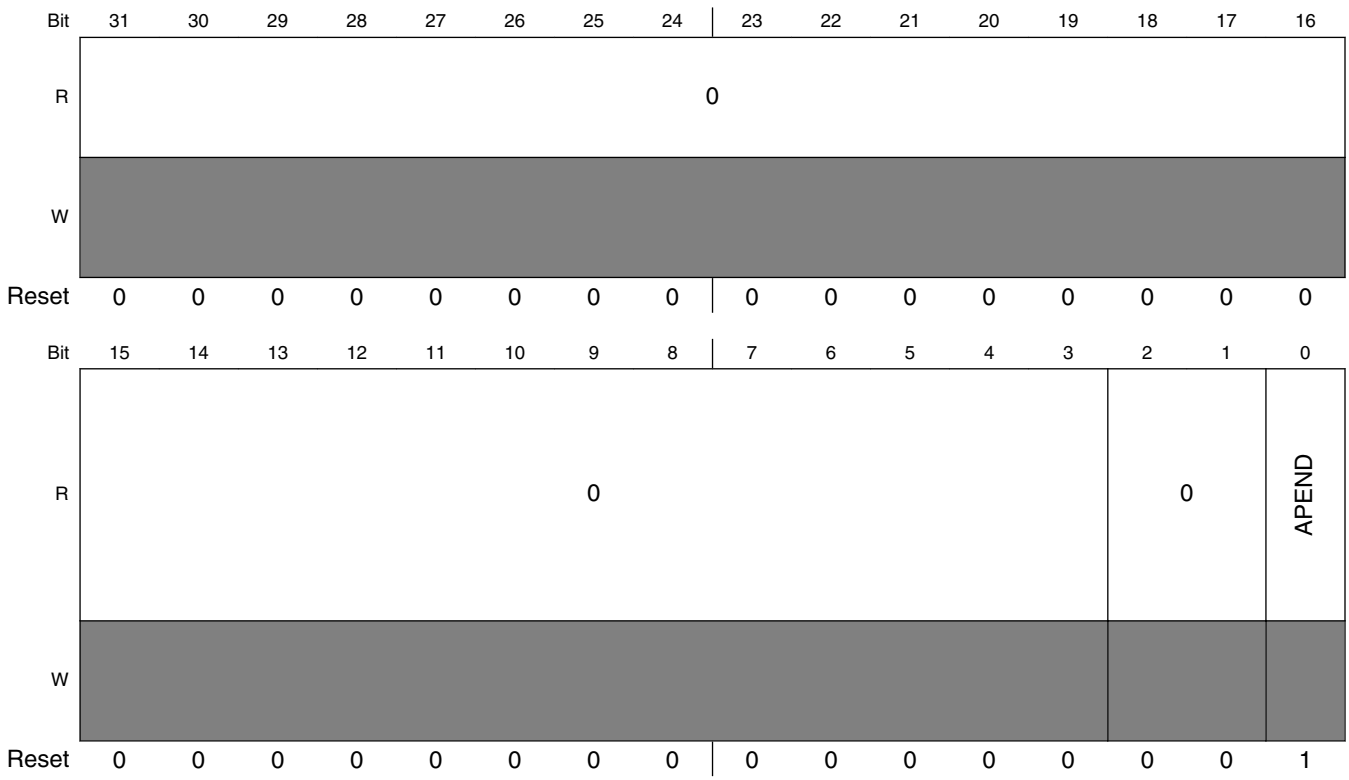


SDMACORE_MCHN0ADDR field descriptions

Field	Description
31–15 Reserved	This read-only field is reserved and always has the value 0.
14 SMSZ	The bit 14 (Scratch Memory Size) determines if scratch memory must be available after every channel context. After reset, it is equal to 0, which defines a RAM space of 24 words for each channel. All of this area stores the channel context. By setting this bit, 32 words are reserved for every channel context, which gives eight additional words that can be used by the channel script to store any type of data. Those words are never erased by the context switching mechanism. 0 24 words per context 1 32 words per context
CHN0ADDR[13:0]	Contains the address of the channel 0 routine programmed by the ARM platform; it is loaded into a general register at the very start of the boot and the SDMA jumps to the address it contains. By default, it points to the standard boot routine in ROM.

55.10.19 ENDIAN Status Register (SDMACORE_ENDIANNESS)

Address: 20E_C000h base + 1Dh offset = 20E_C01Dh

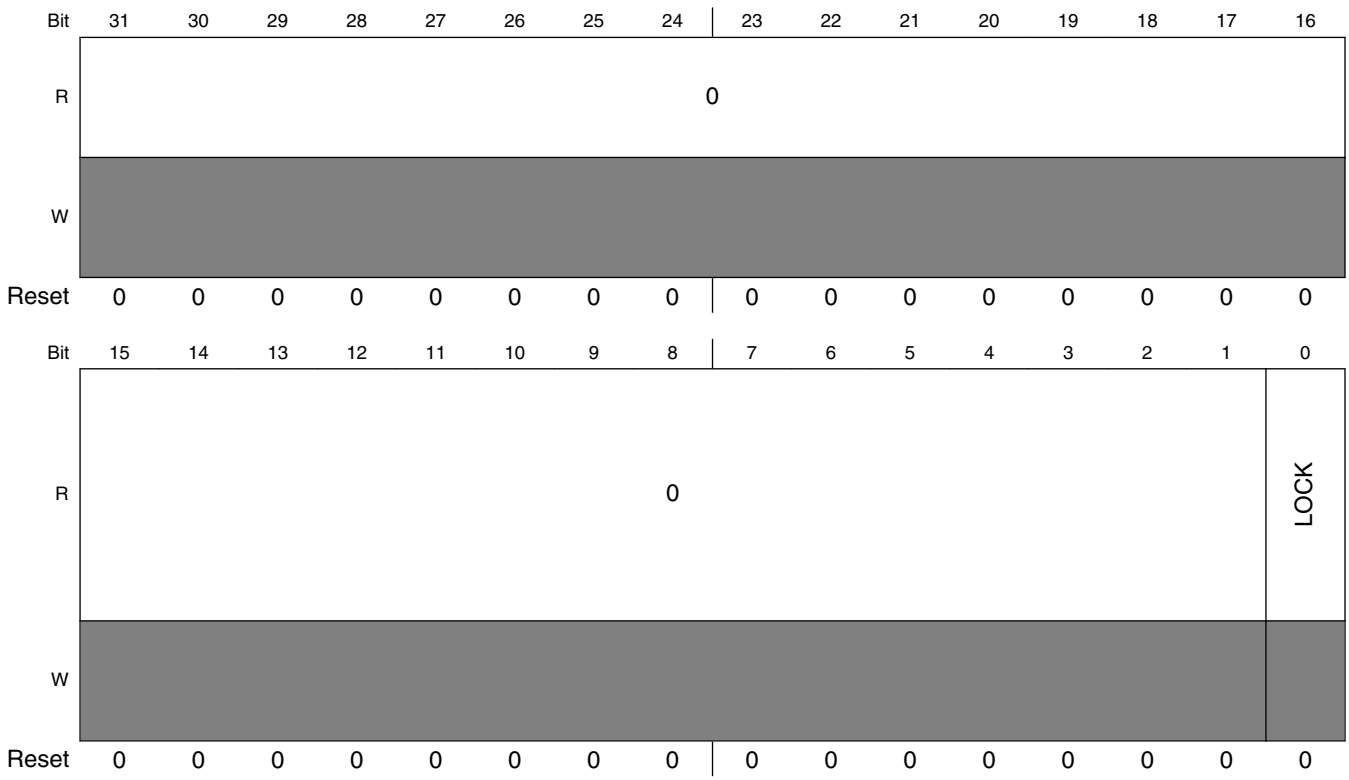


SDMACORE_ENDIANNESS field descriptions

Field	Description
31–3 Reserved	This read-only field is reserved and always has the value 0.
2–1 Reserved	This read-only field is reserved and always has the value 0.
0 APEND	APEND indicates the endian mode of the Peripheral and Burst DMA interfaces. This bit is tied to logic '1' indicating little-endian mode. 0 - ARM platform is in big-endian mode 1 - ARM platform is in little-endian mode

55.10.20 Lock Status Register (SDMACORE_SDMA_LOCK)

Address: 20E_C000h base + 1Eh offset = 20E_C01Eh



SDMACORE_SDMA_LOCK field descriptions

Field	Description
31–1 Reserved	This read-only field is reserved and always has the value 0.
0 LOCK	The LOCK bit reports the value of the LOCK bit in the SDMA_LOCK status register. SDMA software may use this value to determine if certain operations such as loading of new scripts is allowed. 0 - LOCK bit clear 1 - LOCK bit set

55.10.21 External DMA Requests Mirror #2 (SDMACORE_EVENTS2)

Address: 20E_C000h base + 1Fh offset = 20E_C01Fh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																EVENTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMACORE_EVENTS2 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
EVENTS	Reflects the status of the SDMA's external DMA requests. It is meant to allow any channel to monitor the states of these SDMA inputs. This register displays EVENTS 32-47. The separate EVENTS register displays events 0-31.

55.11 SDMA Peripheral Registers

Refer to the respective peripherals' chapters for more information.

Chapter 56
System JTAG Controller (SJC)

56.1 Overview

The System JTAG Controller (SJC) provides debug and test control with the maximum security.

The test access port (TAP) is designed to support features compatible with the IEEE Standard 1149.1 v2001 (JTAG). IEEE P1149.6 standard extensions for AC testing is provided for selected analog IO pads of PCIe and SATA modules.

The figure below shows an overview of the JTAG architecture.

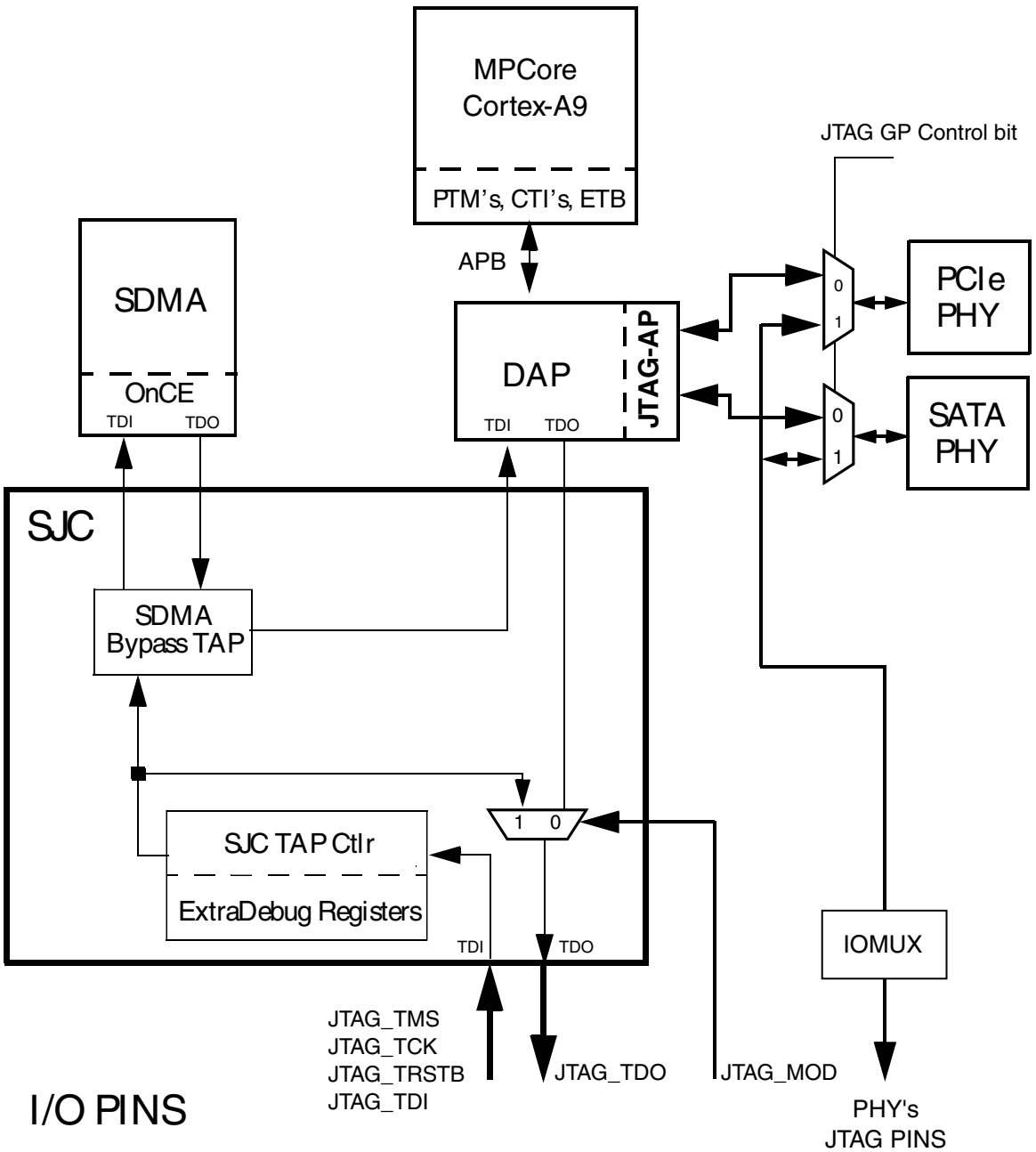


Figure 56-1. System JTAG Controller (SJC) Block Diagram

56.1.1 Features

The System JTAG Controller (SJC) provides the following capabilities:

- JTAG IEEE1149.1 mandatory instructions, see EXTEST Instruction, SAMPLE/PRELOAD Instruction, and BYPASS Instruction.

- JTAG IEEE1149.1 optional instructions, see ID_CODE Instruction (IDCODE), and HIGHZ Instruction.
- JTAG IEEE P1149.1 (standard JTAG) interface to off-chip test and development equipment including an SJC-only mode for true IEEE 1149.1 compliance, used primarily for board-level implementation of boundary scan.
- IEEE P1149.6 (JTAG) mandatory instructions, see EXTEST_PULSE instruction and EXTEST_TRAIN instruction. These two instructions enable edge-detecting behavior on the signal path containing AC pins.
- Debug-related control and status, such as putting selected cores into reset and/or debug mode and the ability to monitor individual core status signals via JTAG.
- Provides means for accessing each OnCE/ICE TAP controller independently to control a target system (see Modes of Operation).
- ExtraDebug logic (see ENABLE_ExtraDebug Instruction).
- The maximum clock speed of the SJC is one-eight of the lowest frequency of the accessed OnCE/ICE. For example in normal operation (no core in low-power mode), this frequency is one-eight of the SDMA frequency if this core is present in the TDI-TDO chain (serially connected with other cores or standalone). The user must also consider the 25 MHz frequency limitation on the CE bus.
- Core compliant modes to support standalone core debuggers (see Modes of Operation).
- Multi-cores daisy chained mode (default one) to support multi-core debuggers (see Modes of Operation).

Detailed information about the SJC is provided in the Security Reference Manual. Contact your Freescale representative for information about obtaining this document.

56.1.2 Modes of Operation

The SJC modes are controlled through both the TAP select register (SJC_TSR) and the MOD input port.

The MOD port (typically connected to pad of the same name) selects between two possible topologies of TAP connections, as seen at SoC level:

- Negating it (this should be the default state) selects all the TAPs (SJC, SDMA and DAP) to be connected in the TDI-TDO chain, which is referred to as "daisy chain" mode, throughout this chapter.
- Asserting it only selects the SJC TAP to be connected in the TDI-TDO chain.

IEEE1149.1 standard features are enabled by configuring the SJC input pin: MOD. Refer to the following table for MOD settings details:

Table 56-1. SJC Modes

MOD	Name	Description
0	Daisy chain ALL	For common SW debug (High speed and production)
1	SJC only	IEEE 1149.1 JTAG compliant mode

The following figure shows the SJC mode selection flow. The numbers shown in parenthesis below each block name indicates the TAP's IR length.

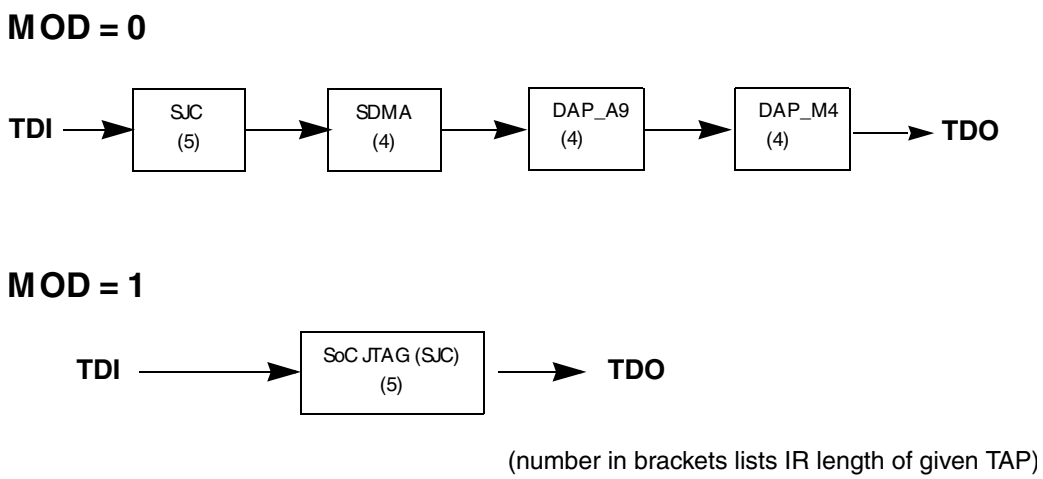


Figure 56-2. SJC Mode Selection Using MOD Pin Sampling

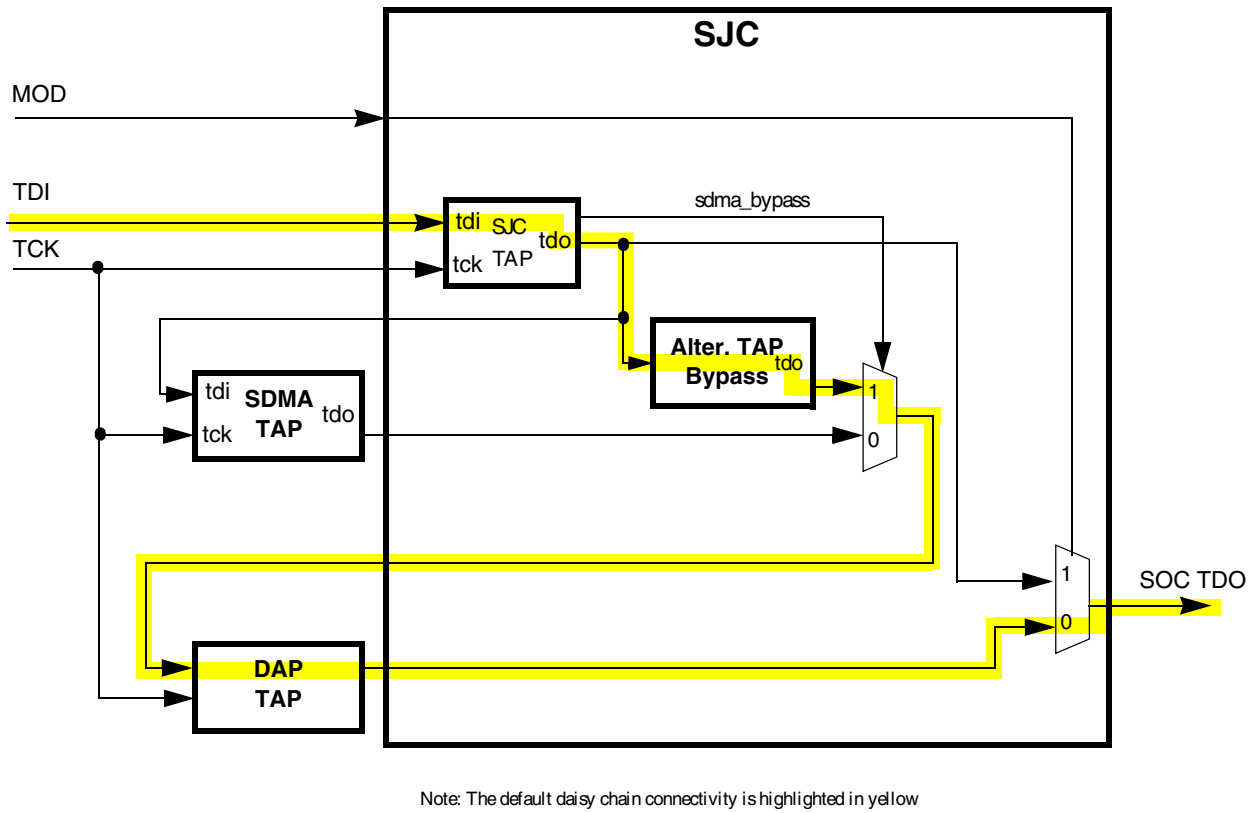
The Connect SDMA bit inside TAP select register controls the SDMA TAP bypass.

- When negated (should be the default state), the SDMA TAP is bypassed with a single D-FF (Flip-flop) during Shift-Dr path
- When asserted SDMA TAP is connected inside the chain
- When taking the SDMA into bypass or out of bypass (by writing to tapsel reg), additional cycle with TMS '0' should be given

The TAP selection block (TSB) provides a simple method of integrating various pieces of IP that have embedded TAPs.

- Provides a way to connect up multiple TAPs within a single SoC
- Identify the SJC TAP as the master TAP which controls the boundary chain (for IEEE 1149.1 standard compliance)
- Follow the state of SJC TAP, and when the Test-Logic-Reset (TLR) state is reached, reset all TAPs

The figure below shows the TAP Selection Block and SOC TAP Chain Scheme.



Note: The default daisy chain connectivity is highlighted in yellow

Figure 56-3. TAP Selection Block and SoC TAP Chain Scheme

NOTE

It is the responsibility of the user to ensure that in any configuration of the TAP controllers chosen, all of the TAPs in the chain comply with the demands of TCK clock frequency as well as the required ratio between TCK clock frequency and that of the core's to which the TAP refers.

56.2 External Signals

The table found here describes the external signals of SJC.

Table 56-2. SJC External Signals

Signal	Description	Pad	Mode	Direction
JTAG_DE_B (DE_B)	SoC debug request/acknowledge pin. The DE_IN_B pin is used to propagate an external debug request event to the core(s). This functionality must be enabled first, by set of DE_to_ARM /	GPIO_16	ALT7	I/O

Table continues on the next page...

Table 56-2. SJC External Signals (continued)

Signal	Description	Pad	Mode	Direction
	DE_to_SDMA bits in SJC's DCR register. It is SoC implementation dependent, whether this pin can also be used to reflect the debug acknowledge event back from the cores (in the case where an Open-Drain scheme is used externally).			
JTAG_MOD (MOD)	SJC mode selection. This pin is sampled at TRST reset to determine two possible modes for the TAP connection configuration.	JTAG_MOD	No Muxing	I
JTAG_TCK (TCK)	Test Clock (TCK). This is used to synchronize the test logic and includes an internal pull-up resistor	JTAG_TCK	No Muxing	I
JTAG_TDI (TDI)	Test Data Input (TDI). Serial test instruction and data are received through the test data input (TDI) pin. TDI is sampled on the rising edge of TCK and includes an internal pullup resistor	JTAG_TDI	No Muxing	I
JTAG_TDO (TDO)	Test Data Output (TDO). The serial output for test instructions and data. TDO is tri-statable and is actively driven in the shift-IR and shift-DR controller states. TDO changes on the falling edge of TCK	JTAG_TDO	No Muxing	O
JTAG_TMS (TMS)	Test Mode Select (TMS). This is used to sequence the test controller's state machine. TMS is sampled on the rising edge of TCK and includes an internal pullup resistor	JTAG_TMS	No Muxing	I
JTAG_TRSTB (TRSTB)	Test Reset (TRST). This is used to asynchronously initialize the test controller. The TRST pin has an internal pullup resistor	JTAG_TRSTB	No Muxing	I

56.2.1 External Signal Overview

The SJC provides test and debug control with a minimum number of contacts.

The figure below shows SJC connections to external contacts and other chip blocks.

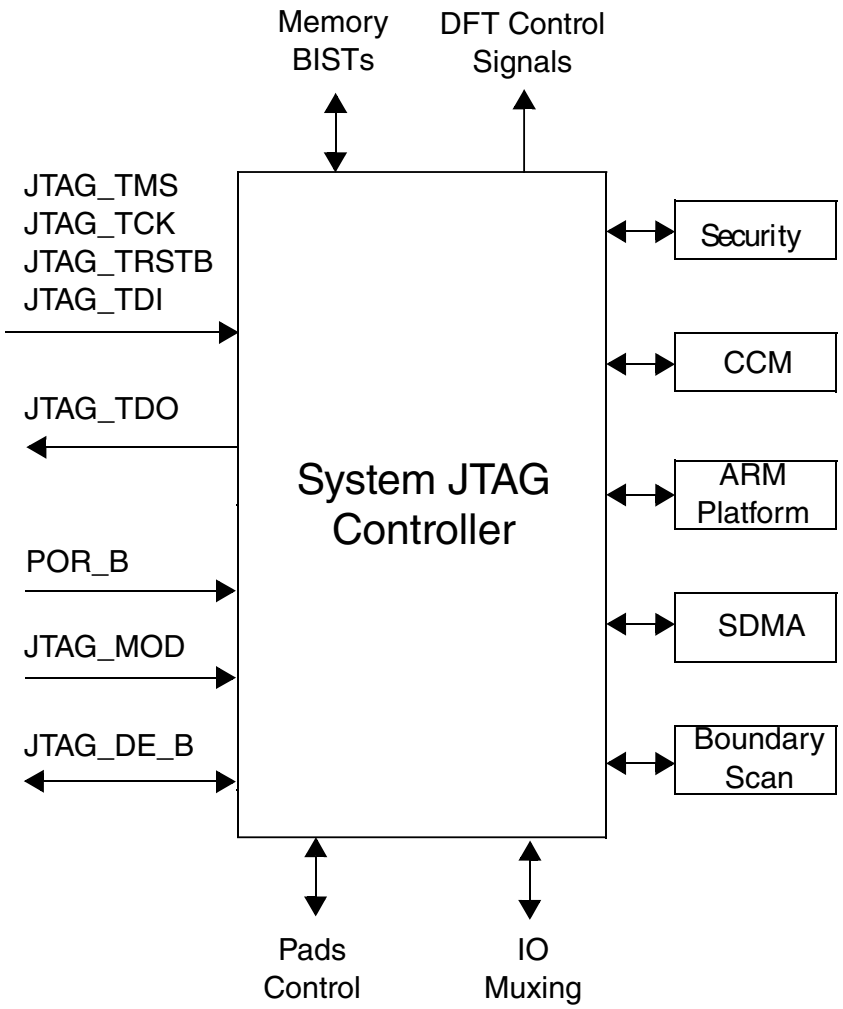


Figure 56-4. SJC Connections

56.2.2 TAP Controller

The TAP controller is responsible for interpreting the sequence of logical values on the TMS signal. It is a synchronous state machine that controls the operation of the JTAG logic. The value shown adjacent to each arc represents the value of the TMS signal sampled on the rising edge of TCK signal. For a description of the TAP controller states, refer to the appropriate IEEE 1149.1 document.

The state machine is shown in the following figure.

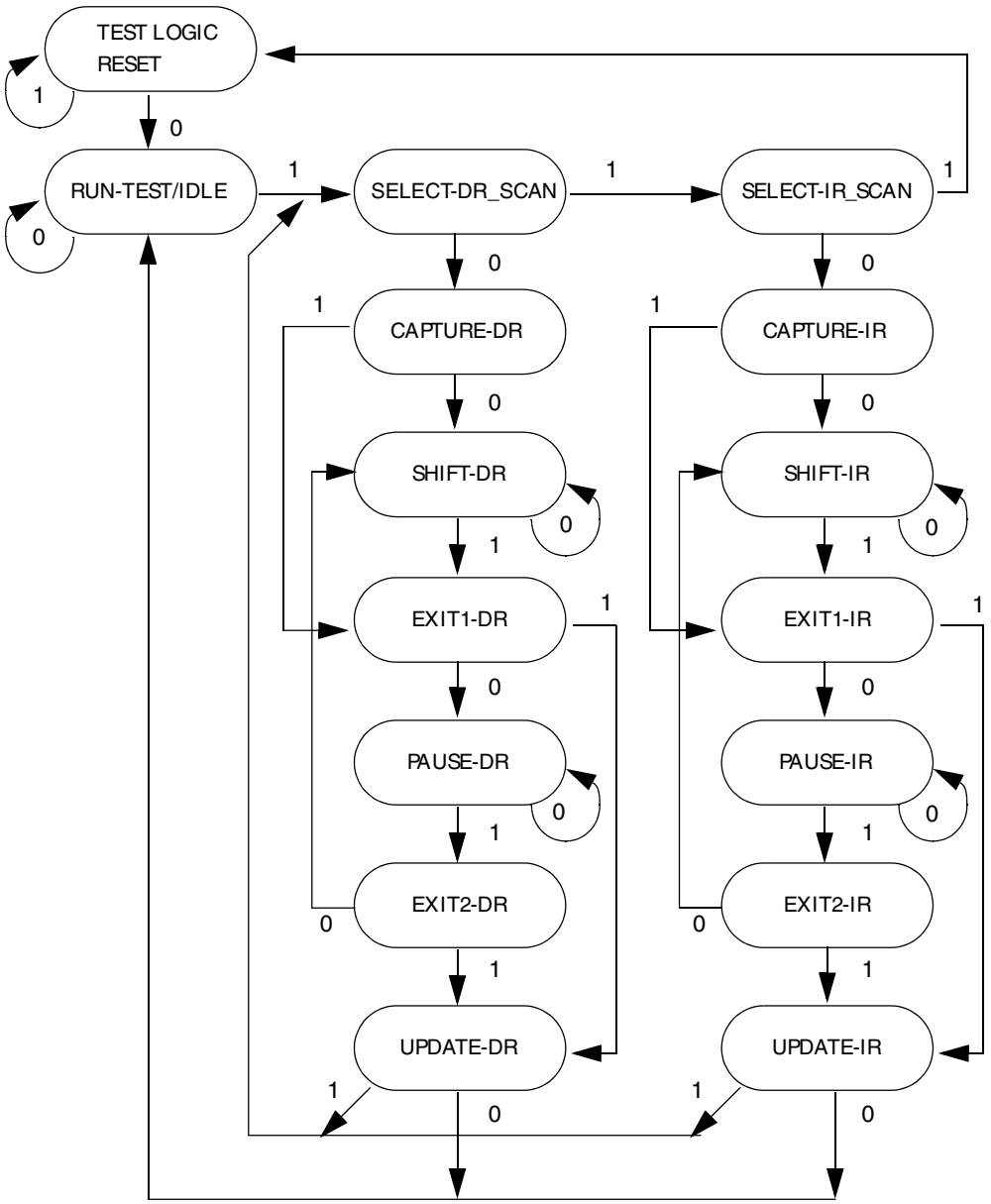


Figure 56-5. TAP Controller State Machine

The change of the JTAG state machine occurs on the rising edge of TCK. TMS and TDI change on the falling edge of TCK. TDO also changes on the falling edge of TCK following entry into the Shift_DR or Shift_IR states (TDO_EN is the enable of the tristate buffer driving the TDO output).

The figure below shows the timings of the SJC signals.

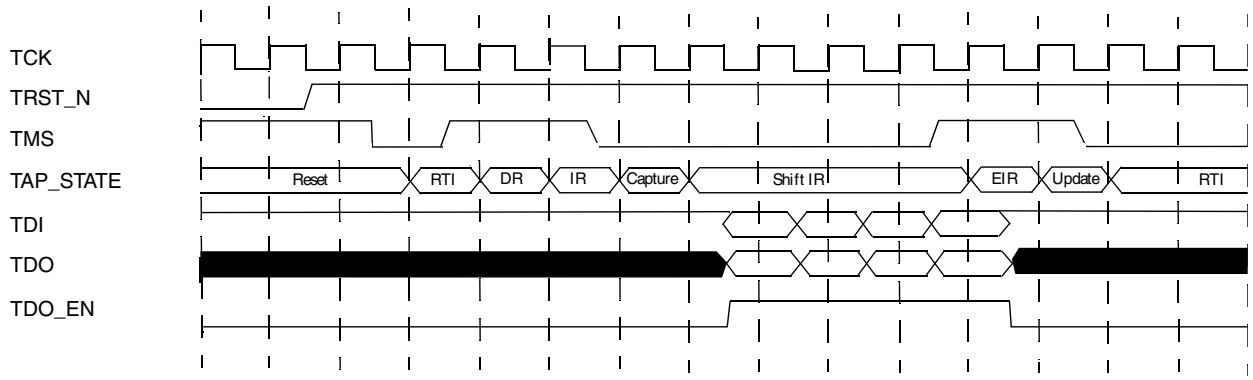


Figure 56-6. SJC Signals Timing Diagram

56.2.3 Accessing ExtraDebug Registers

Accessed through the Select-DR-Scan path, the ExtraDebug shift register consists of 38 bits (maximum) comprising a 32-bit data field (max length, see extradebug register description), a 5 bit address field and read/write bit.

The write actually takes place when the JTAG TAP controller enters the Update-DR state. On a read, the data field is ignored (the user should shift only 5 times to enter Read=1 and the address), the read takes place on the next path through DR at the Capture-DR state, the data is shifted-out during the Shift-DR state.

On the second path for a read access, simultaneous write access is not supported: command converter software shifts in zeros so the TAP decodes a write to the CSR (read-only register) which does not have any effect on the circuit.

The number of shift depends on the width of the accessed register as explained in the following diagrams.

First a write access (one path through Select-DR-Scan):

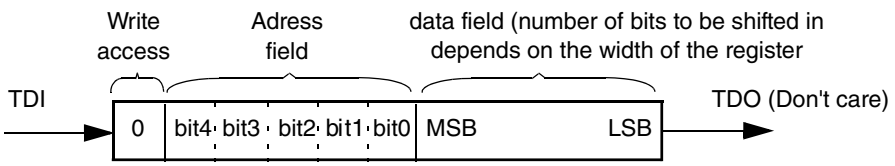


Figure 56-7. TDI/TDO on write access

Then a read access (requires two paths through Jtag DR Scan path):

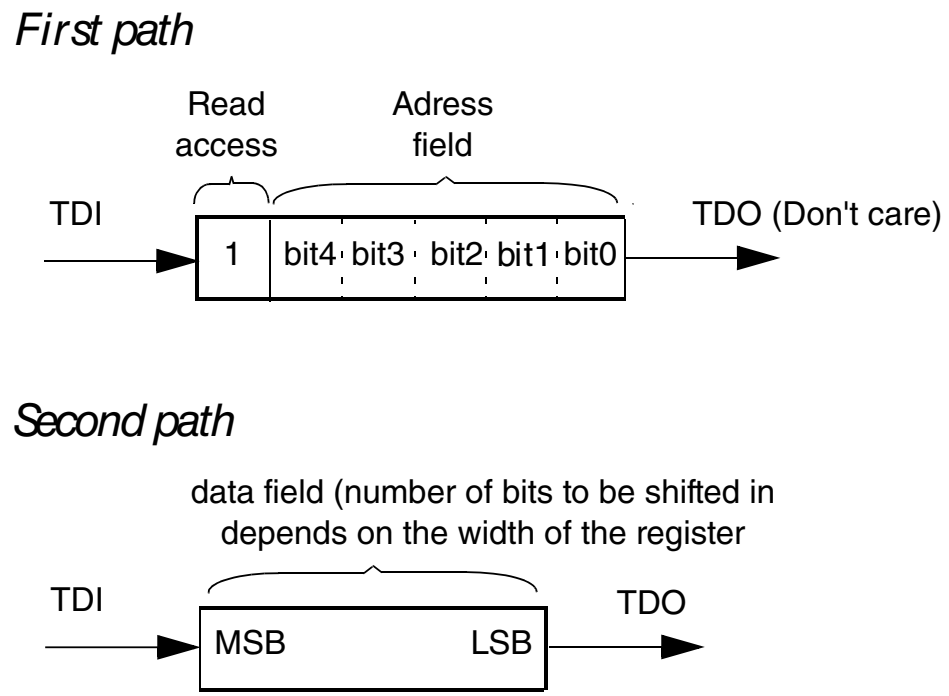


Figure 56-8. TDI/TDO on Read Access

For example, write value 0b1010_1100 to Debug Control Register (address = 0b00110).

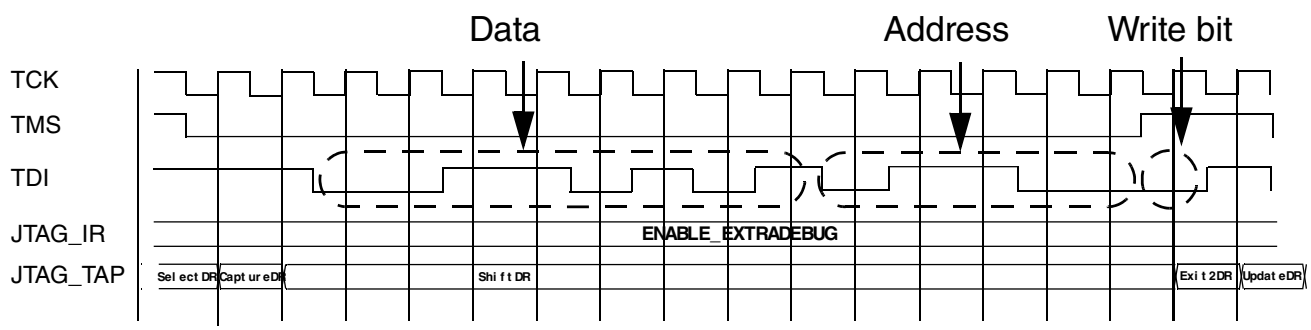


Figure 56-9. Example: Write Access to DCR

The SJC registers have different levels of security (refer to JTAG Security Modes):

- Secured- accessible only in mode 2 (supposed correct response entered), mode 3 and mode 4.
- Unsecured- accessible in all modes

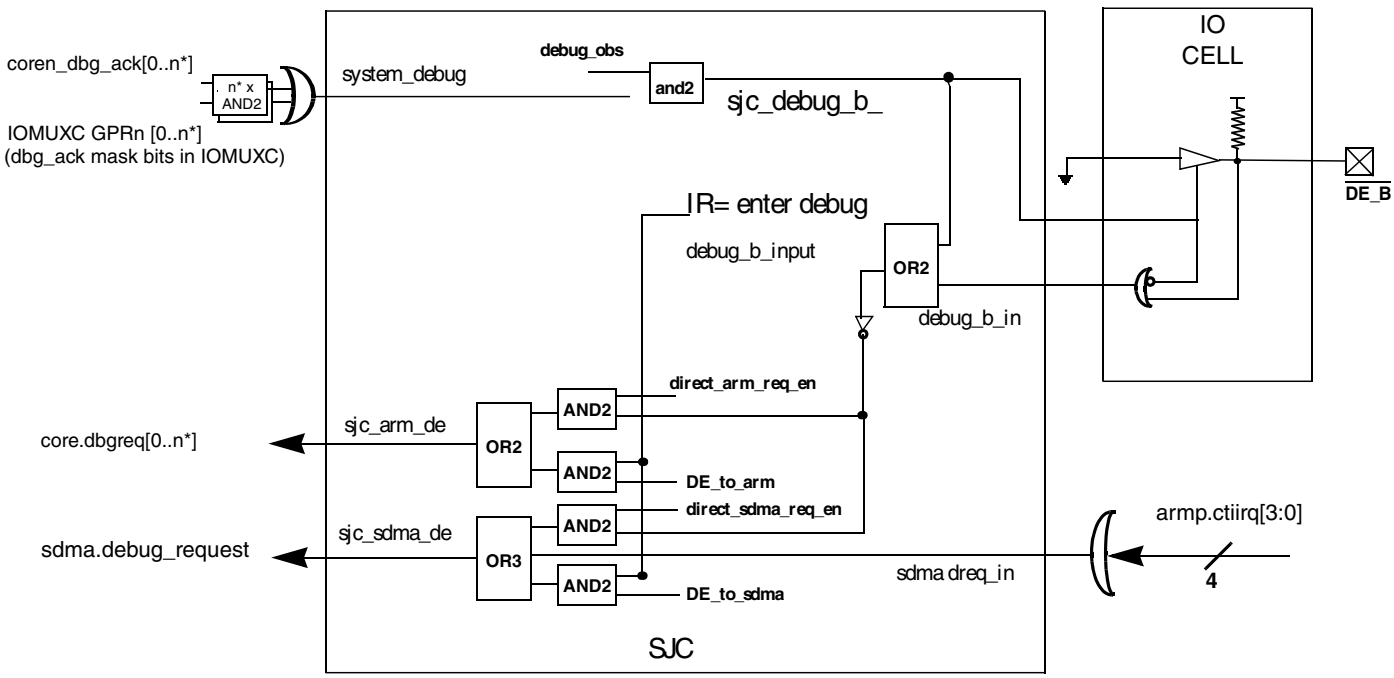
The level of security of each register is indicated in its name or description, in "Programmable Registers" section.

A single DE_B pin is dedicated for debug request input/output in bidirectional open drain functionality (including an internal pull-up device).

Bits 6:5 in DCR register serve as mask bits, controlling the propagation of external debug request to each recipients (ARM Platform, SDMA).

The bits 1:0 define the propagation enable of IR debug request to recipient cores.

The following figure shows the $\overline{DE}$ Pin Select Logic.



* "n" - denotes the number of cores in a specific SoC.

Figure 56-10. $\overline{DE}$ Pin Select Logic

For security reasons, bits for output and input propagation control are at their negated values after reset. A user cannot put the cores in debug mode through $\overline{DE}$ without any Jtag access.

The configuration after reset prevents propagation of debug requests / acknowledges to or from the cores.

56.3 TAP Selection Block (TSB)

As described in Modes of Operation, the SJC can access cores in different modes selected through a TSB.

56.3.1 Select Mode Using Software

Conceptually, the SJC_TSR is a data register which is accessed through Access TSR IR instruction of SJC TAP.

The following figure shows the process of using reserved IR to access the SJC_TSR.

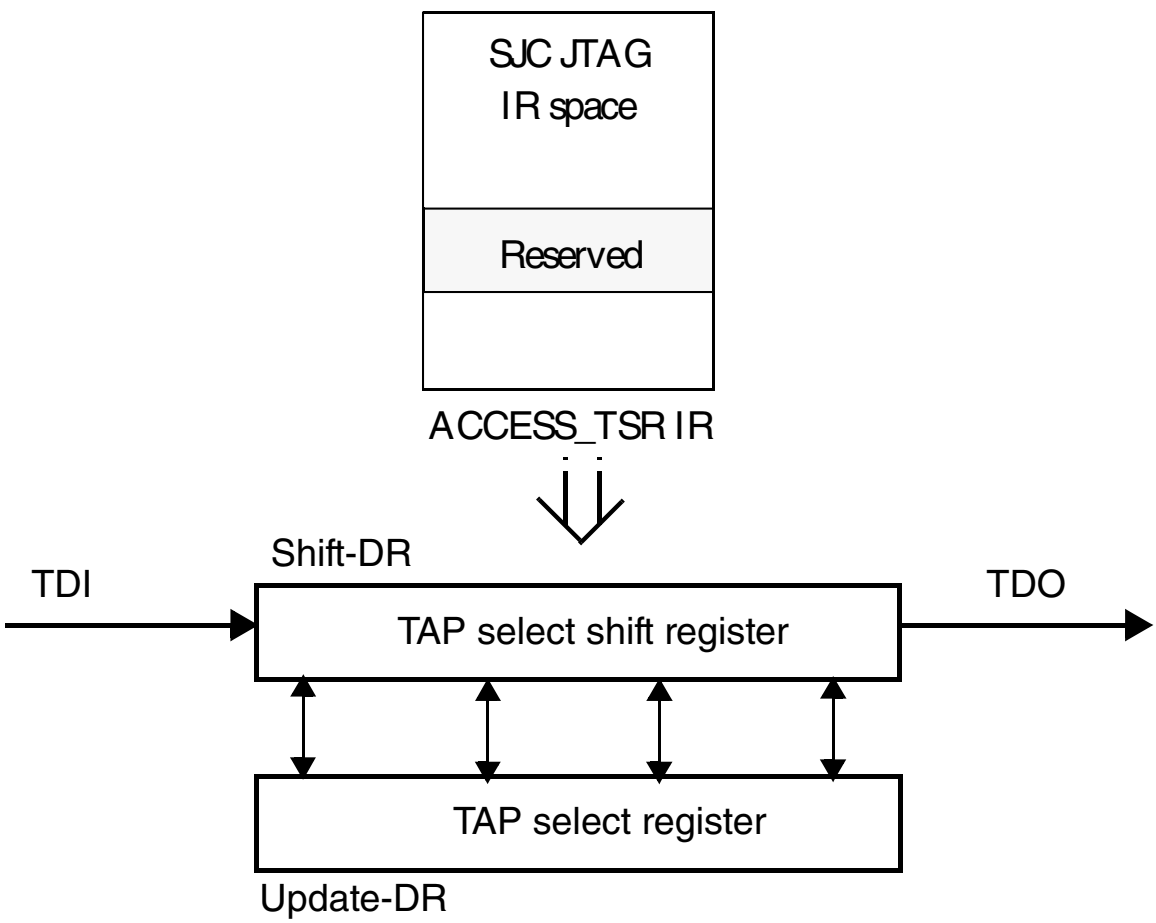


Figure 56-11. Using Reserved IR to Access the TAP Select Register (SJC_TSR)

The SJC_TSR can only be changed during the update-DR state of the TSB JTAG state machine. This is necessary to prevent a TAP that is being selected from losing synchronization with the TSB state machine when the TSB state machine returns to run-test-idle. Therefore, an associated shift register for the SJC_TSR is loaded into the

SJC_TSR during the update-DR state (see the figure above). The shift register must also capture the state of the SJC_TSR when in the Capture-DR state for visibility of the contents of the SJC_TSR. See TAP Select Instruction, for more information.

56.4 Boundary Scan Register (BSR)

The Boundary Scan Register (BSR) in the JTAG implementation contains bits for all device signal and clock pins and associated control signals.

All SoC bidirectional pins have a single register bit in the boundary scan register for pin data, and are controlled by an associated control bit in the boundary scan register.

56.5 SoC JTAG Instruction Register (SJIR)

The SoC JTAG Instruction register is 5 bits wide.

Table 56-3. SoC JTAG Instruction Register (SJIR)

Code					SJC IR
B4	B3	B2	B1	B0	
0	0	0	0	0	IDCODE
0	0	0	0	1	SAMPLE/PRELOAD
0	0	0	1	0	EXTEST
0	0	0	1	1	HI-Z
0	0	1	0	0	ENABLE_ExtraDebug
0	0	1	0	1	ENTER_DEBUG (secured)
0	0	1	1	0	Reserved
0	0	1	1	1	TAP select
0	1	0	0	0	EXTEST_PULSE
0	1	0	0	1	EXTEST_TRAIN
0	1	0	1	0	Reserved
0	1	0	1	1	Reserved
0	1	1	0	0	Security Output challenge
0	1	1	0	1	Security Enter response
-					Reserved
1	1	1	1	1	BYPASS

The instruction register is reset to 0b00000 in the test-logic-reset controller state which is equivalent to the IDCODE instruction.

During the capture-IR controller state, the parallel inputs to the instruction register are loaded with the code 01 in the least significant bits as required by the standard; the most significant bits are loaded with the values 00, leading to a capture value of 0b00001.

56.5.1 ID_CODE Instruction (IDCODE)

Selects the ID register, and the system logic controls the I/O pins. This instruction is provided as a public instruction to allow the manufacturer, part number and version of a component to be determined through the TAP.

The table below shows the ID register configuration.

Table 56-4. ID Configuration Register (IDCODE)

IDCODE				ID Configuration Register												
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	Version Information[3:0]				Part Number (Bits 27-16)											
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x
Note:																
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	Part Number (Bits 15-12)				Manufacturer Identity											1
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	x	x	x	x	0	0	0	0	0	0	0	1	1	1	0	1
Note:																

Table 56-5. ID Configuration Register Description (IDCODE)

Field	Description
31-28 Version Information	IC/SoC Version information number. Initial value: '0000' This number is subject to changes, for new IC/SoC (System On A Chip) revision releases.
27-12 Part Number	Customer Part Number The 16-bit Part Number value is unique for every Freescale's SoC / IC. See "System Debug" chapter for exact register value for a specific SoC.
11-1 Manufacturer Identity	Manufacturer Identity Freescale's Manufacturer Identity code. Bits [11:1] - 00000001110
0	Tied to logic 1.

One application of the ID register is to distinguish the manufacturer(s) of components on a board when multiple sourcing is used. As more components emerge which conform to the IEEE 1149.1 standard, it is desirable to allow for a system diagnostic controller unit to blindly interrogate a board design to determine the type of each component in each location. This information is also available for factory process monitoring and for failure mode analysis of assembled boards.

Once the IDCODE instruction is decoded, it selects the ID register which is a 32 Bit data register. Because the bypass register loads a logic 0 at the start of a scan cycle, whereas the ID register loads a logic 1 into its least significant bit, examination of the first bit of data shifted out of a component during a test data scan sequence immediate following exit from Test-Logic-Reset controller state shows whether such a register is included in the design. When the IDCODE instruction is selected, the operation of the test logic has no effect on the operation of the on-chip system logic as required by the IEEE 1149.1 standard.

56.5.2 SAMPLE/PRELOAD Instruction

Selects the boundary scan register and the system logic controls the I/O pins.

The SAMPLE/PRELOAD instruction provides two separate functions:

- First, it provides a means to obtain a snapshot of system data and control signals. The snapshot occurs on the rising edge of TCK in the capture-DR controller state. The data can be observed by shifting it transparently through the boundary scan register.
- The second function of SAMPLE/PRELOAD is to initialize the boundary scan register output cells prior to selection of EXTEST. This initialization ensures that known data appears on the outputs when entering the EXTEST instruction.

NOTE

Because there is no internal synchronization between the JTAG clock (TCK) and the system clock (CLK), the user must provide some form of external synchronization to achieve meaningful results.

For more details on the function and use of SAMPLE/PRELOAD, refer to the appropriate IEEE 1149.1 document.

56.5.3 EXTEST Instruction

Selects the boundary scan register, and the 1149.1 test logic has control of the I/O pins.

By using the TAP controller, the register is capable of:

- Scanning user-defined values into the output buffers,
- Capturing values presented to input pins
- Controlling the direction of bidirectional pins,
- Controlling the output drive of tri-statable output pins.

For more details on the function and use of EXTEST, refer to the appropriate IEEE 1149.1 document.

The EXTEST instruction also asserts internal reset for the cores (through CCM, refer to Figure 56-14) to force a predictable internal state while performing external boundary scan operations.

56.5.4 HIGHZ Instruction

All output drivers, including the two-state drivers, are turned off (that is, high impedance). The instruction selects the bypass register.

In this mode, all internal pullup resistors on all the pins (except for the TMS, TDI, TCK, TRSTB pins) are disabled. This disabling functionality is not built into SJC, but should be implemented by some logic in the SOC/IO Pads.

For more details on the function and use of HIGHZ, refer to the IEEE 1149.1 document.

The HIGHZ instruction also asserts internal reset for the cores (through CCM, refer to Figure 56-14) to force a predictable internal state while performing external boundary scan operations.

56.5.5 BYPASS Instruction

Selects the single Bit bypass register and the system logic controls the I/O pins.

This creates a shift-register path from TDI to the bypass register and, finally, to TDO, circumventing the boundary scan register. This instruction is used to enhance test efficiency when a component other than the SoC Core based device becomes the device under test.

When the bypass register is selected by the current instruction, the shift-register stage is set to a logic zero on the rising edge of TCK in the capture-DR controller state. Therefore, the first bit to be shifted out after selecting the bypass register is always a logic zero.

For more details on the function and use of BYPASS, refer to the appropriate IEEE 1149.1 document.

56.5.6 ENABLE_ExtraDebug Instruction

The TDI and TDO pins are connected directly to the ExtraDebug registers, the SJC TAP controller remaining connected to TDI and TMS.

The ExtraDebug shift register consists of 38 bits (maximum) comprising a 32-bits data field (maximum length, see Accessing ExtraDebug Registers,), a 5 bits address field and read/write bit. On a register read, the data field does not need to be filled in. The particular ExtraDebug register connected between TDI and TDO at a given time is selected by the ExtraDebug controller depending on the ExtraDebug Address being currently decoded. All communication with the ExtraDebug controller is done through the Select-DR-Scan path of the JTAG TAP Controller.

56.5.7 ENTER_DEBUG instruction

The ENTER_DEBUG instruction is used to generate a debug request event to SDMA and the ARM MPCore Platform simultaneously (practically, inherited minimal skew is expected, due to difference in event signal propagation in the different modules).

The TDI and TDO are connected to the Instruction Register (IR). After the acknowledgment of the Debug Mode is received (can be checked by reading the Core Status Register part of the ExtraDebug logic), the user can perform system debug functions on the cores.

NOTE

The ENTER_DEBUG event issue to the cores, can be masked, by bits in DCR register.

It is user's responsibility to shift-in another IR value (like IDCODE) before trying to bring the cores out of debug mode, as the debug request signals to the cores remains asserted as long as ENTER_DEBUG IR is in place.

The user need to check that cores are in debug mode (watching debug acknowledge signal) before leaving ENTER_DEBUG instruction, otherwise debug request might not take affect.

56.5.8 TAP Select Instruction

By means of TAP select instruction a user can access TAP select register and by controlling its only bit SDMA Bypass, control whether SDMA TAP is bypassed or not.

Table 56-6. TAP Select Register (TSR)

	TAP Select Register
	BIT 0
	Connect SDMA
TYPE	rw
RESET	0
Note:	

Table 56-7. TAP Select Register Description

Field	Description
0 SDMA Bypass	Connect SDMA Control whether SDMA TAP is bypassed or not: • 0 - SDMA TAP is bypassed by the alternate TAP inside SJC (emulating 4-bit IR and 1-bit bypass path). • 1 - SDMA TAP is connected to the TDI-TDO chain. **NOTE:** Additional cycle with TMS '0' should be inserted, after writing to this register, to allow the SDMA tap be sync before SDMA get into / out of bypass.

56.5.9 EXTEST_PULSE instruction

The EXTEST_PULSE instruction implements new test behaviors for AC pins and simultaneously behaves identically to IEEE Std 1149.1 EXTEST for DC pins.

56.5.10 EXTEST_TRAIN instruction

The EXTEST_TRAIN instruction implements new test behaviors for AC pins and simultaneously behaves identically to IEEE Std 1149.1 EXTEST for DC pins.

56.6 Security

JTAG manipulation is one of the known hackers' ways of executing unauthorized program code, getting control over the OS and run code in privileged modes.

The SJC provides a debug access to several H/W blocks including the ARM processor and the system bus. This allows for program control and manipulation as well as visibility into system peripherals and memory. The PTM interface allow bus transactions to be traced. Together these tools provide the hacker all the access needed to completely comprise the system. Means must be provided to block any malicious JTAG access.

The SJC provides a way of regulating the JTAG access.

The following are the different JTAG security modes:

- Mode #1: No Debug-Maximum Security. All security sensitive JTAG features are permanently blocked.
- Mode #2: Secure JTAG-High security. JTAG use is regulated by secret key based authentication mechanism.
- Mode #3: JTAG Enabled-Low security. JTAG always enabled.

The JTAG security modes are configured using eFUSEs which can burned after packaging by applying electrical signals. The fuse burning is irreversible process, once a fuse is burned (e-fuse or laser fuse) it is impossible to change the fuse back to the un-burned state.

56.6.1 JTAG Security Modes

JTAG can be in one of JTAG security modes which is selected by setting the SJC eFUSE configuration. The physical location of the fuses is not in the SJC.

56.6.1.1 Mode 1: No Debug - Maximum Security

No Debug JTAG security mode provides the highest security level.

In this mode, all JTAG features are disabled except for:

- ScanBoundary Scan
- MBIST, all modes except for debug modes which enable controlled memory contents output
- PLL BIST
- BIST monitor mode, allowing routing to external pins BIST pass/fail/invoke information
- PLL bypass- Bypass ARM or/and USB PLL.
- Visibility of the following status bits: power mode - normal, standby, stop, shutdown, and so on

These features do not reduce the security level of the product, and they allows to perform important tests and board connectivity checks.

56.6.1.2 Mode 2: Secure JTAG - High Security

The Secure JTAG mode limits the JTAG access by using challenge/response based authentication mechanism. Any access to JTAG port is being checked. Only authorized debug devices (that is, devices having the right response) can access the JTAG, unauthorized JTAG access attempts are denied.

The intent of this mode is to allow return field testing. When a secured JTAG device is being returned for debugging, this mode allows authorized re-activation of the JTAG.

56.6.1.2.1 Challenge/Response Mechanism in System JTAG Mode

When SJC is in Sysytem JTAG mode the authentication process is as follows:

1. Shift Output Challenge instruction to IR.
2. Passing through Capture-DR state of the SJC and by performing Shift-DR operations Challenge code can be accessed from TDO.
3. Shift Enter Response instruction to IR. By performing Shift-DR, operations enter Response code value through TDI. As Update-DR state is entered, Response code is compared with the correct one.

In Fixed challenge-response pair mode, each part has its individual challenge - response pair which is determined at manufacturing time, and does not change later on. The SJC compares the user's response to the expected response.

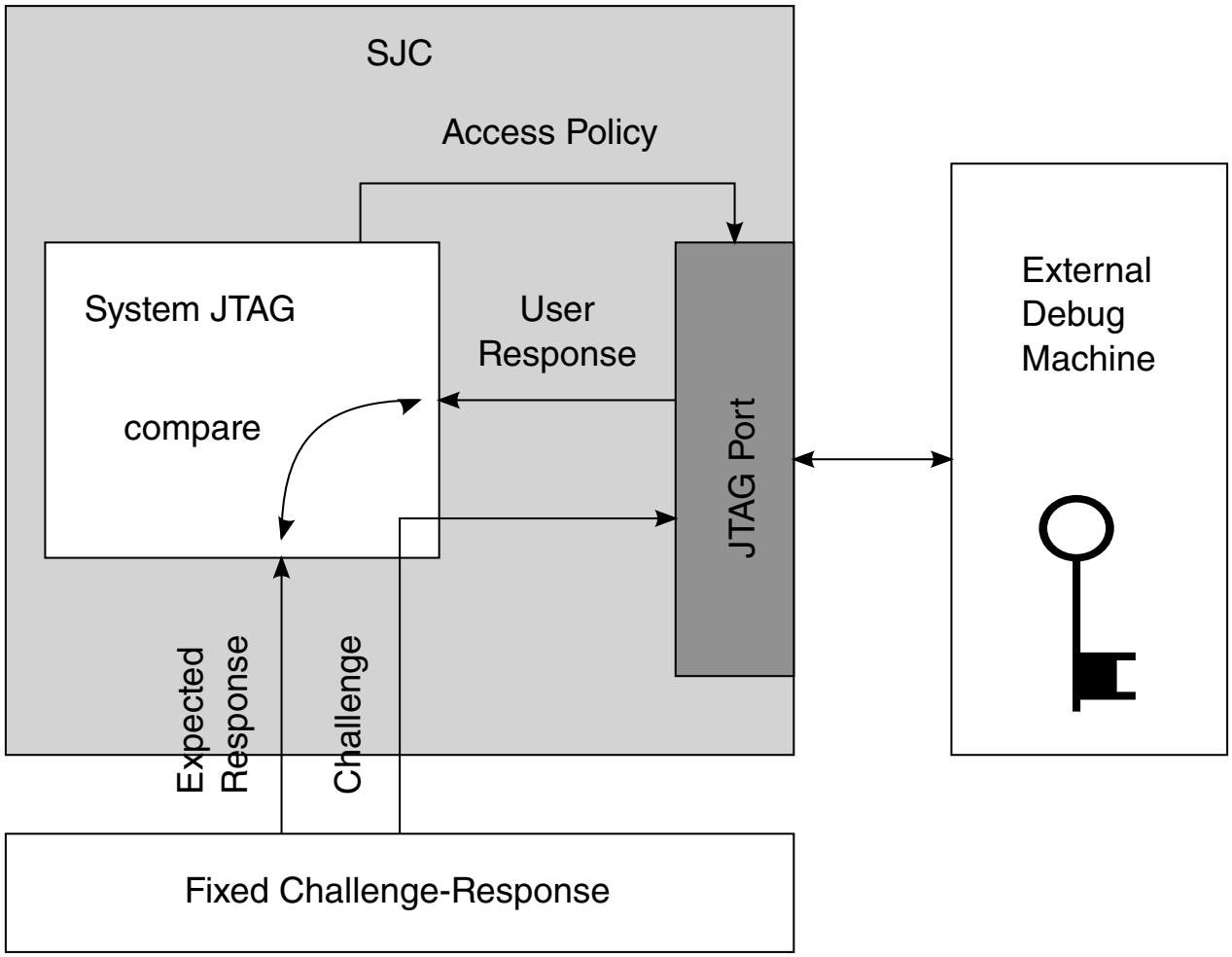


Figure 56-12. Mode #2 - Secure JTAG with Fixed Challenge-response Pair

56.6.1.3 Mode 3: JTAG Enabled - Low Security

In the JTAG Enabled JTAG security mode, all JTAG features are enabled.

56.6.2 Software Enabled JTAG

To increase the flexibility of the SJC, an option to enable the JTAG via software is added and is available only in Secure JTAG mode. By writing '1' to HAB_JDE (HAB JTAG DEBUG ENABLE) bit in the e-fuse controller module (OCOTP_CTRL), the JTAG is opened, regardless of its security mode. It is the responsibility of software to assert or negate this bit.

Additionally, a corresponding lock bit is available (in the e-fuse control module) to ensure that only trusted software is able to set the JDE bit. When the LOCK bit is set, no future change of JDE is possible, until the next POR (power-on-reset) cycle.

The platform initialization software should set the LOCK bit for JDE bit before transferring control to the application code.

The S/W JTAG enable allows JTAG enabling without activating the challenge-Response mechanism (which requires JTAG access tool enhancement or special H/W). The JTAG S/W enable does not allow debug in case of boot or memory fault as it requires reset before entering debug.

This feature can be permanently blocked by burning the dedicated e-fuse.

NOTE

The S/W enabled JTAG feature reduces the overall security level of the system as it relies on S/W protections. If this feature is not required, it is strongly recommended to burn the JTAG_HEO e-fuse which disables this feature.

56.6.3 Kill Trace

The kill trace signal disables any output of the PTM block. The PTM can be accessed either via JTAG port and/or by direct software code. Blocking the JTAG port also yields assertion of the kill trace signal. This resulted in blocking of trace port. The intention of this action is to block any attempt to break into the system via software manipulation of the debug modules. The kill trace, when active, prevents trace output even in case where it can be activated via chip pin.

The kill trace feature needs to be activated by burning a dedicated e-fuse. If the fuse is left intact, kill trace is never activated as seen in Figure 56-13.

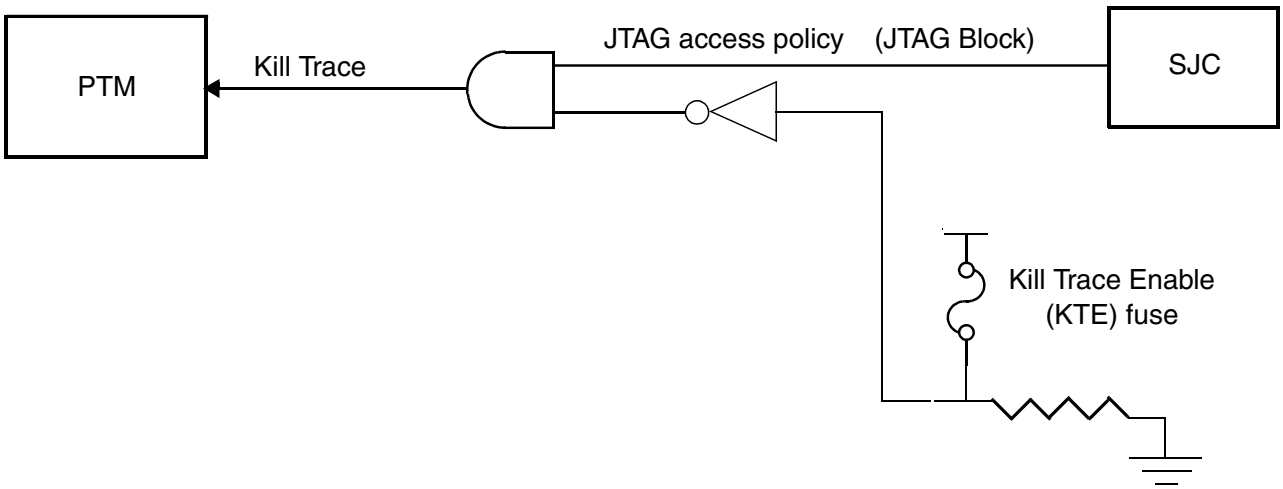


Figure 56-13. Kill Trace eFUSE

The kill trace is asserted when "kill trace enable" fuse is burned and "ipt_secur_block" signal in SJC is asserted, which happens when at least one of the following is true:

- Mode #2 (Secure JTAG) and no code has been entered
- Mode #2 (Secure JTAG) with incorrect response entered
- Mode #1 (No debug)
- TRST_B signal is active
- POR has not ever been asserted

56.6.4 SJC Disable Fuse

In addition to the different JTAG security modes that are implemented internally in the System JTAG Controller (SJC), there is an option to disable the SJC functionality by eFUSE configuration. This creates additional JTAG mode that is, JTAG Disabled with highest level of JTAG protection. In this mode all JTAG features are disabled.

Specifically, the following debug features are disabled in addition to the features that were already disabled in No Debug JTAG mode:

- Memory BIST
- Boundary scan register (SJC_BSR)
- Non-Secure JTAG control registers (PLL configuration, Deterministic Reset, PLL bypass)
- Non-Secure JTAG status registers (Core status)
- Chip Identification Code (IDCODE)

56.7 Functional Description

This section provides a complete functional description of the block.

56.7.1 Static Core Debug

The SJC JTAG TAP controller is fully compatible with the IEEE 1149.1a-2001 Standard Test Access Port and Boundary Scan Architecture specifications.

The ARM MPCore platform debug system (named CoreSight) including the real-time Program Trace Macrocell (PTM), are controlled via the Debug Access Port (DAP) module. Refer to ARM MPCore and PTM Technical reference manuals for more details.

The SDMA has a TAP controller to manage its own OnCE, see SDMA OnCE specifications for more details.

The OnCE and ICE provide a mean of interacting with the cores and their peripherals non-intrusively so that a user may examine registers, memories to facilitate hardware and software development. Refer to TAP Selection Block (TSB), for more information.

56.7.2 Reset Mechanism

The following figure shows the SJC reset logic

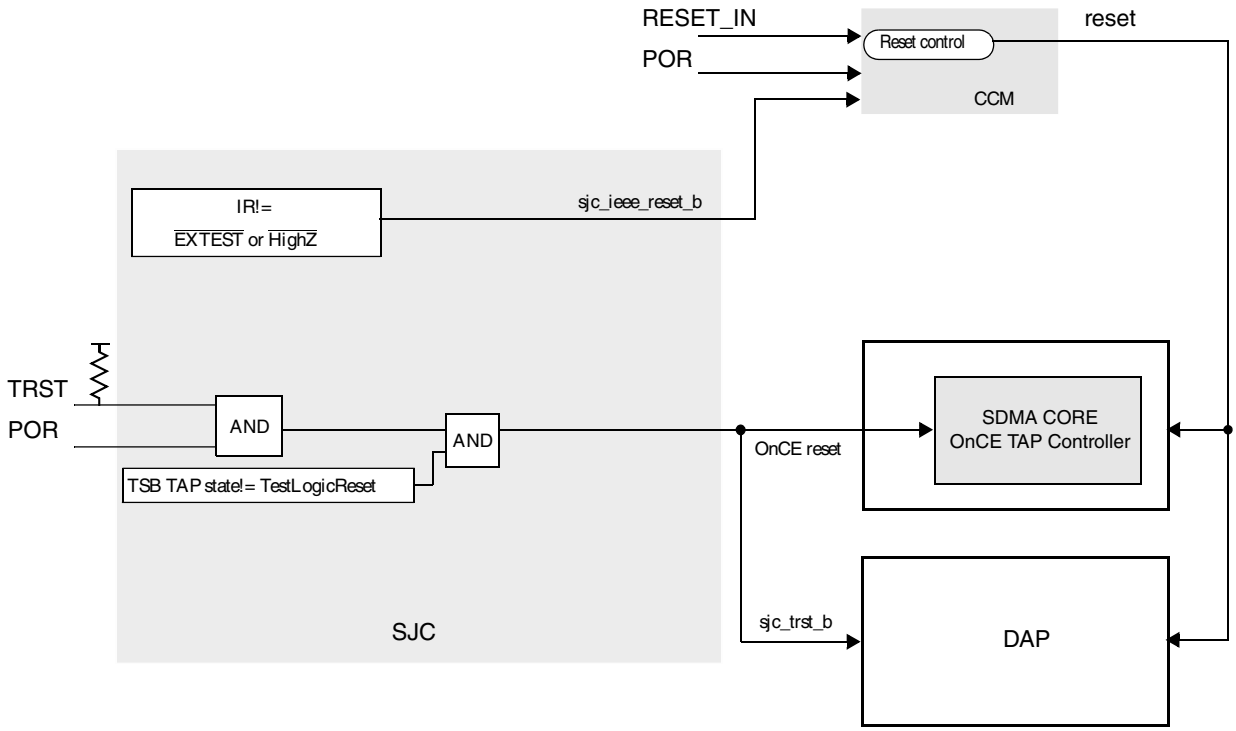


Figure 56-14. SJC Reset Logic

NOTE

- Asserting TRSTB in any scan mode resets the TCR loosing the testmode configuration and selects default TAP.
- SJC generates an IEEE reset signal to the CCM when in one of the IEEE modes HIGHZ or EXTEST. This signal generates a system reset to the cores until exit from one of these modes.
- The TSB generates Once/ICE reset (either TRSTB if implemented or other) when its TAP state reaches Test-Logic-Reset (meaning that TAP accessed is also reaching Test-Logic-Reset).

56.8 Initialization/Application Information

The control afforded by the output enable signals using the boundary scan register and the EXTEST instruction requires a compatible circuit-board test environment to avoid device-destructive configurations. The user must avoid situations in which the SJC output drivers are enabled into actively driven networks.

There are two constraints related to the JTAG interface:

- Ensure that the JTAG test logic is kept transparent to the system logic by forcing TAP into the Test-Logic-Reset controller state. During power-up, SJC's internal TRSTB is asserted as IC's POR_B is asserted which forces the TAP controller into this state. After that, if TMS either remains unconnected or is connected to VCC, then the TAP controller cannot leave the Test-Logic-Reset state, regardless of the state of TCK.
- DE_B is an IO pin with pullup and care must be taken of the direction when driving this signal.

56.9 SJC Memory Map/Register Definition

In addition to the standard accessible JTAG registers (per IEEE1149.1 standard) listed in SoC JTAG Instruction Register (SJIR) , the chip contains the following registers accessed using the ExtraDebug mechanism, controlled via "ENABLE_ExtraDebug" IR instruction.

NOTE

SJC registers are only accessible by JTAG interface. They are not memory mapped to processor address space, so the absolute addresses provided by default in the SJC memory map are not valid.

This section assumes the JTAG controller is accessed in standalone mode or daisy chained (defined by TAP Selection Block) using the appropriate TSB configuration.

See "System Debug" chapter for more details about the general purpose register descriptions that are unique to this chip.

SJC memory map

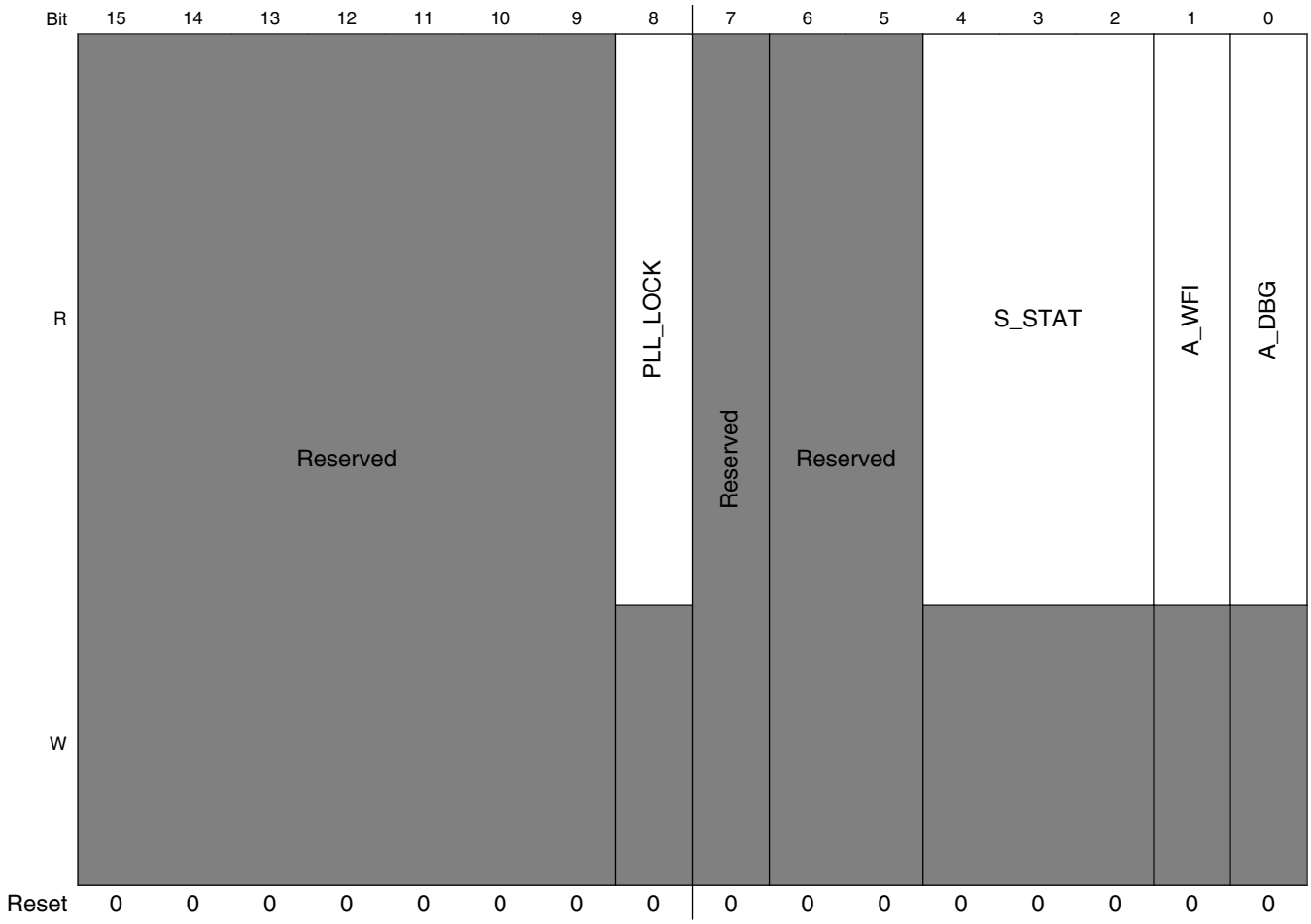
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	General Purpose Unsecured Status Register 1 (SJC_GPUSR1)	32	R	0000_0000h	56.9.1/4959
1	General Purpose Unsecured Status Register 2 (SJC_GPUSR2)	32	R	0000_0000h	56.9.2/4961
2	General Purpose Unsecured Status Register 3 (SJC_GPUSR3)	32	R	0000_0000h	56.9.3/4961
3	General Purpose Secured Status Register (SJC_GPSSR)	32	R	0000_0000h	56.9.4/4962
4	Debug Control Register (SJC_DCR)	32	R/W	0000_0000h	56.9.5/4963
5	Security Status Register (SJC_SSR)	32	R	See section	56.9.6/4965
7	General Purpose Clocks Control Register (SJC_GPCCR)	32	R/W	0000_0000h	56.9.7/4968

56.9.1 General Purpose Unsecured Status Register 1 (SJC_GPUSR1)

The General Purpose Unsecured Status Register 1 is a read only registers used to check the status of the different Cores and of the PLL. The rest of its bits are for general purpose use.

Address: 0h base + 0h offset = 0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

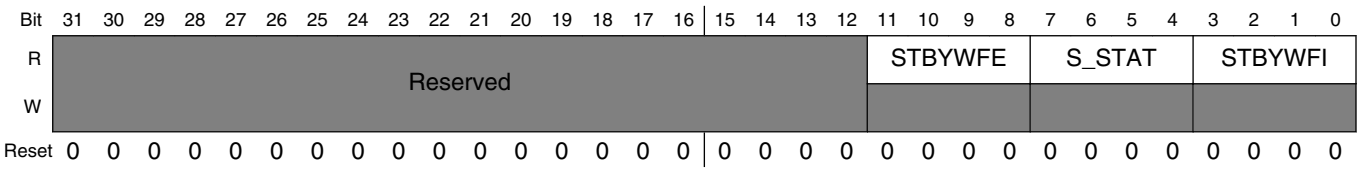


SJC_GPUSR1 field descriptions

Field	Description
31–9 -	This field is reserved. Reserved.
8 PLL_LOCK	PLL_LOCK A Combined PLL-Lock flag indicator, for all the PLL's.
7 -	This field is reserved. Reserved
6–5 -	This field is reserved. Reserved.
4–2 S_STAT	3 LSBits of SDMA core statusH.
1 A_WFI	ARM core wait-for interrupt bit Bit 1 is the ARM core standbywfi (stand by wait-for interrupt). When this bit is HIGH, ARM core is in wait for interrupt mode.
0 A_DBG	ARM core debug status bit Bit 0 is the ARM core DBGACK (debug acknowledge) DBGACK can be overwritten in the ARM core DCR to force a particular DBGACK value. Consequently interpretation of the DBGACK value is highly dependent on the debug sequence. When this bit is HIGH, ARM core is in debug.

56.9.2 General Purpose Unsecured Status Register 2 (SJC_GPUSR2)

Address: 0h base + 1h offset = 1h

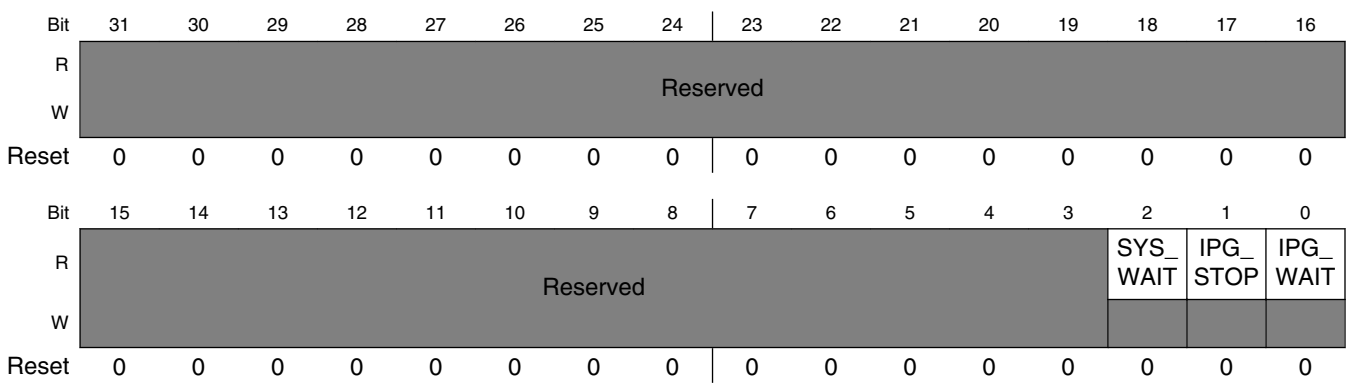


SJC_GPUSR2 field descriptions

Field	Description
31–12 -	This field is reserved. Reserved
11–8 STBYWFE	STBYWFE[3:0] Reflecting the "Standby Wait For Event" signals of all cores.
7–4 S_STAT	S_STAT[3:0] SDMA debug status bits: debug_core_state[3:0]
STBYWFI	STBYWFI[3:0] These bits provide status of "Standby Wait-For-Interrupt" state of all ARM cores.

56.9.3 General Purpose Unsecured Status Register 3 (SJC_GPUSR3)

Address: 0h base + 2h offset = 2h



SJC_GPUSR3 field descriptions

Field	Description
31–3 -	This field is reserved. Reserved

Table continues on the next page...

SJC_GPUSR3 field descriptions (continued)

Field	Description
2 SYS_WAIT	System In wait Indication on System in wait mode (from CCM).
1 IPG_STOP	IPG_STOP CCM's "ipg_stop" signal indication
0 IPG_WAIT	IPG_WAIT CCM's "ipg_wait" signal indication

56.9.4 General Purpose Secured Status Register (SJC_GPSSR)

The General Purpose Secured Status Register is a read-only register used to check the status of the different critical information in the SoC. This register cannot be accessed in secure modes.

Address: 0h base + 3h offset = 3h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	GPSSR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

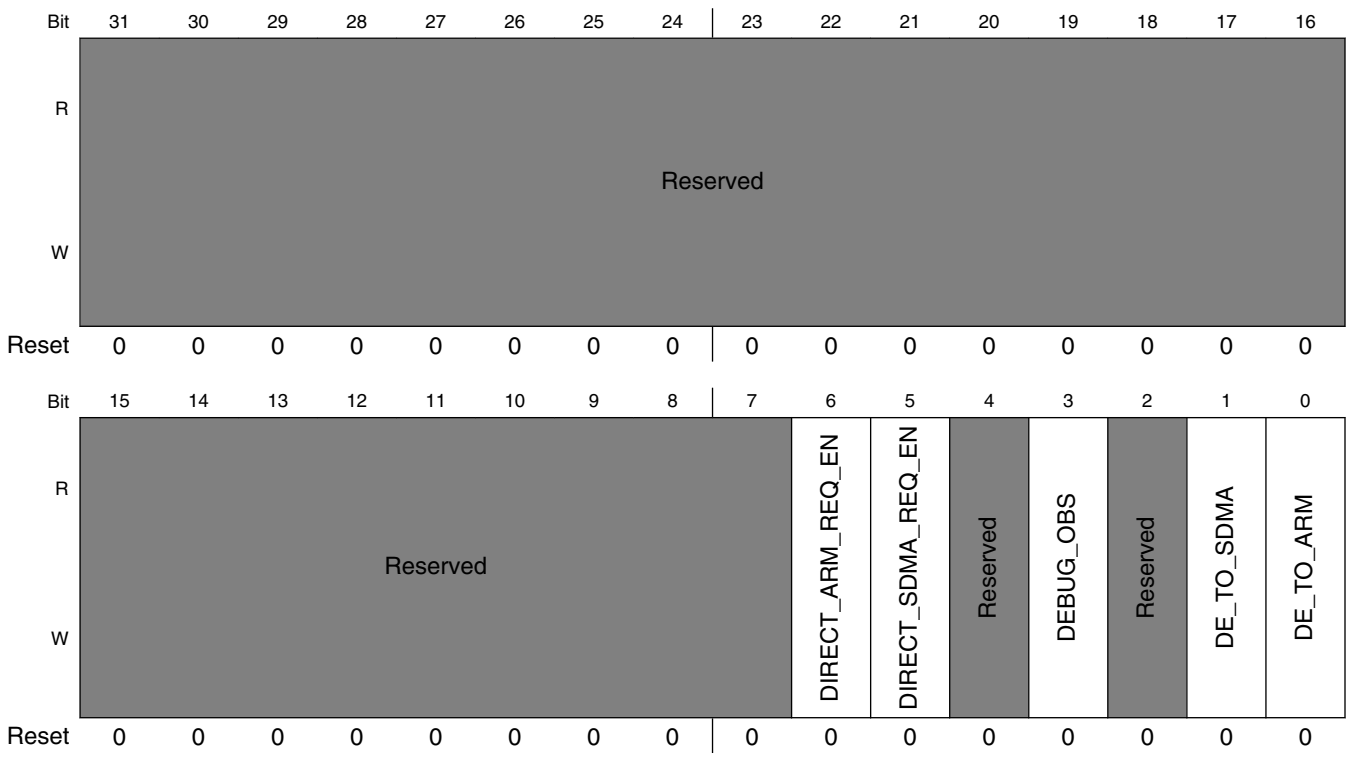
SJC_GPSSR field descriptions

Field	Description
GPSSR	General Purpose Secured Status Register Register is used for testing and debug.

56.9.5 Debug Control Register (SJC_DCR)

This register is used to control propagation of debug request from DE_B pad to the cores and debug signals from internal logic to the DE_B pad.

Address: 0h base + 4h offset = 4h



SJC_DCR field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6 DIRECT_ARM_REQ_EN	Pass Debug Enable event from DE_B pin to ARM platform debug request signal(s). This bit controls the propagation of debug request DE_B to the Arm platform. 0 Disable propagation of system debug to (DE_B pin) to Arm platform. 1 Enable propagation of system debug to (DE_B pin) to Arm platform.
5 DIRECT_SDMA_REQ_EN	Debug enable of the sdma debug request This bit controls the propagation of debug request DE_B to the sdma. 0 Disable propagation of system debug to (DE_B pin) to sdma. 1 Enable propagation of system debug to (DE_B pin) to sdma.
4 -	This field is reserved. Reserved

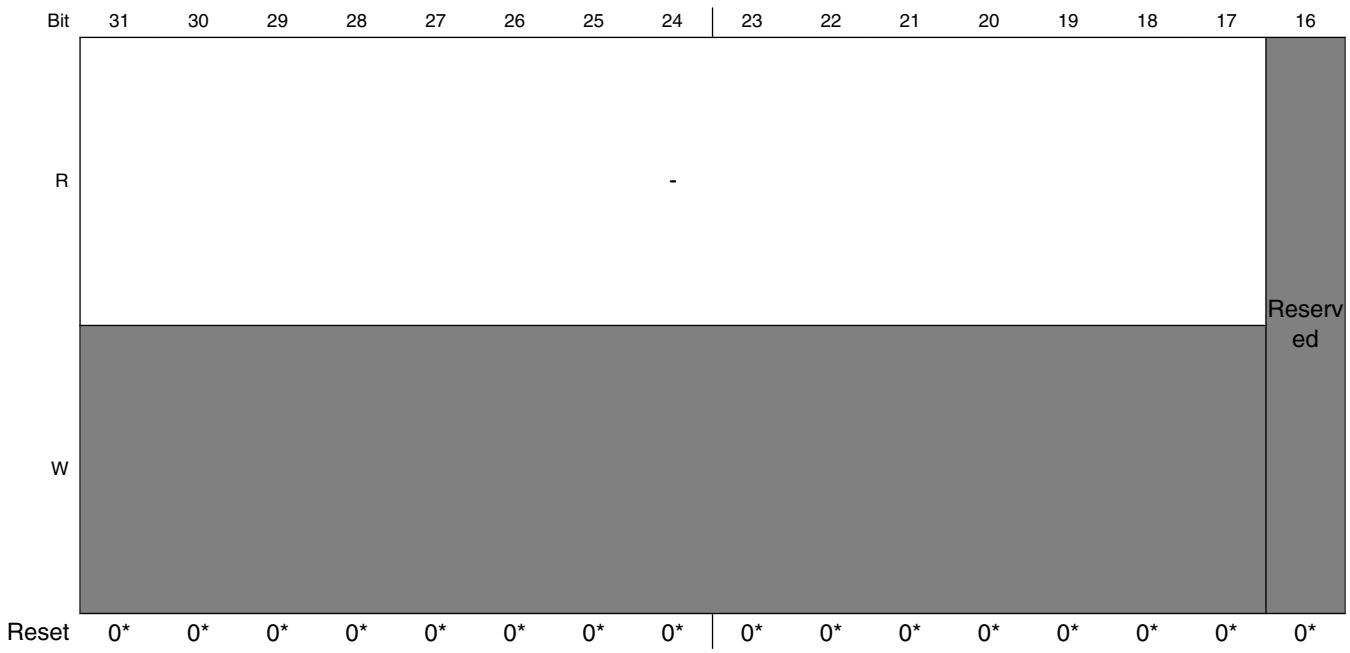
Table continues on the next page...

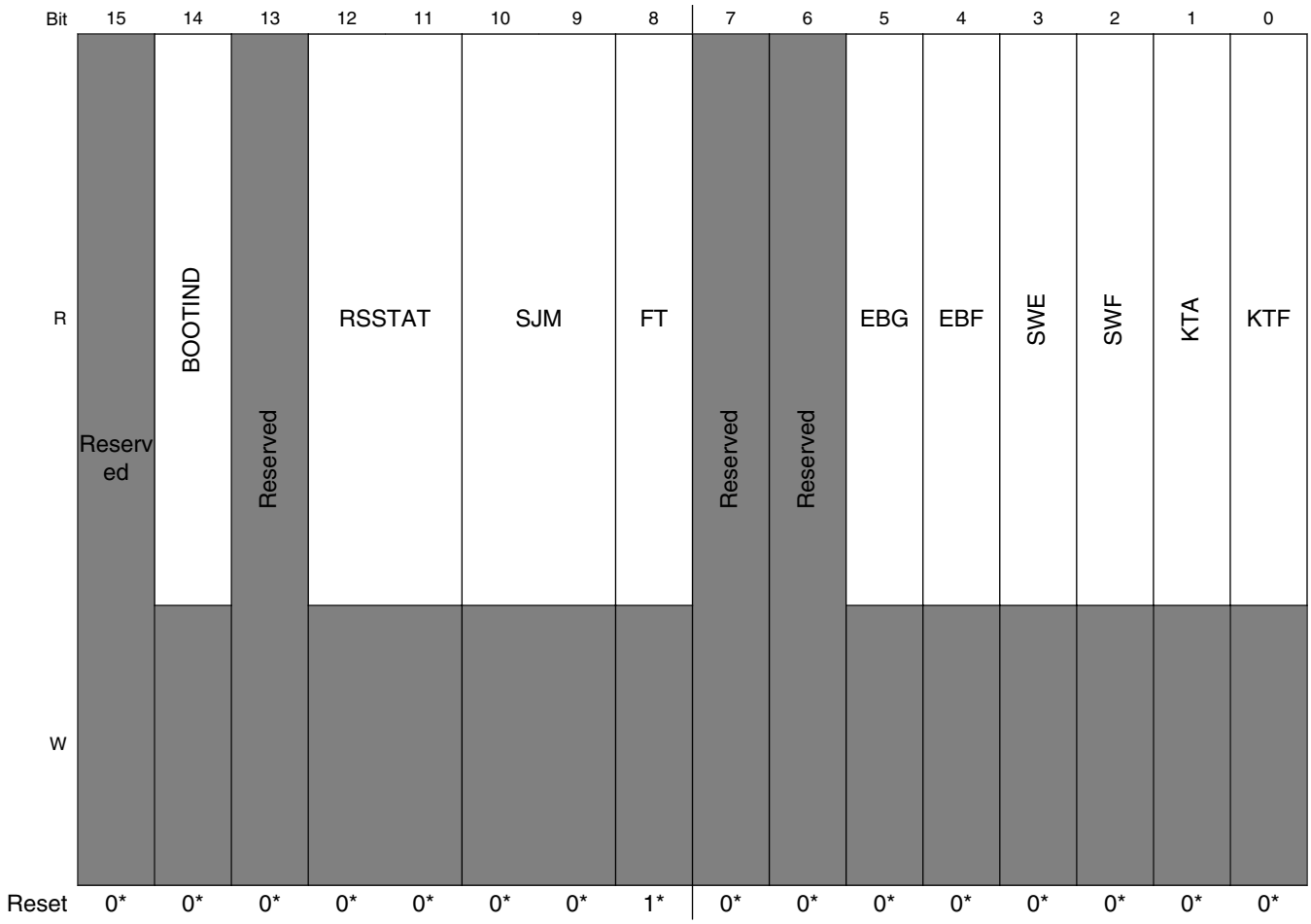
SJC_DCR field descriptions (continued)

Field	Description
3 DEBUG_OBS	Debug observability This bit controls the propagation of the "system debug" input to SJC For i.MX 6x, the SJC's "system_debug" input is tied to logic HIGH value, therefore, set of "debug_obs" bit, will result in unconditional assertion of DE_B pad. 0 Disable propagation of system debug to DE_B pin 1 unconditional assertion of pad. DE_B
2 -	This field is reserved. Reserved
1 DE_TO_SDMA	SDMA debug request input propagation This bit controls the propagation of debug request to SDMA, when the JTAG state machine is put in "ENTER_DEBUG" IR instruction.. 0 Disable propagation of debug request to SDMA 1 Enable propagation of debug request to SDMA
0 DE_TO_ARM	ARM platform debug request input propagation This bit controls the propagation of debug request to ARM platform ("dbgreq"), when the JTAG state machine is put in "ENTER_DEBUG" IR instruction. 0 Disable propagation of debug request to ARM platform 1 Enable propagation of debug request to ARM platform

56.9.6 Security Status Register (SJC_SSR)

Address: 0h base + 5h offset = 5h





* Notes:
- The SJM reset value, reflects the JTAG security state, as defined by status of JTAG_SMODE[1:0] fuses. See the SJM bitfield description for details on valid values.

SJC_SSR field descriptions

Field	Description
31–17 -	Reserved.
16–15 -	This field is reserved. Reserved
14 BOOTIND	Boot Indication Inverted Internal Boot indication, i.e inverse of SRC: "src_int_boot" signal
13 -	This field is reserved. Reserved
12–11 RSSTAT	Response status Response status bits 00 Response wasn't entered 01 Response was entered but not verified

Table continues on the next page...

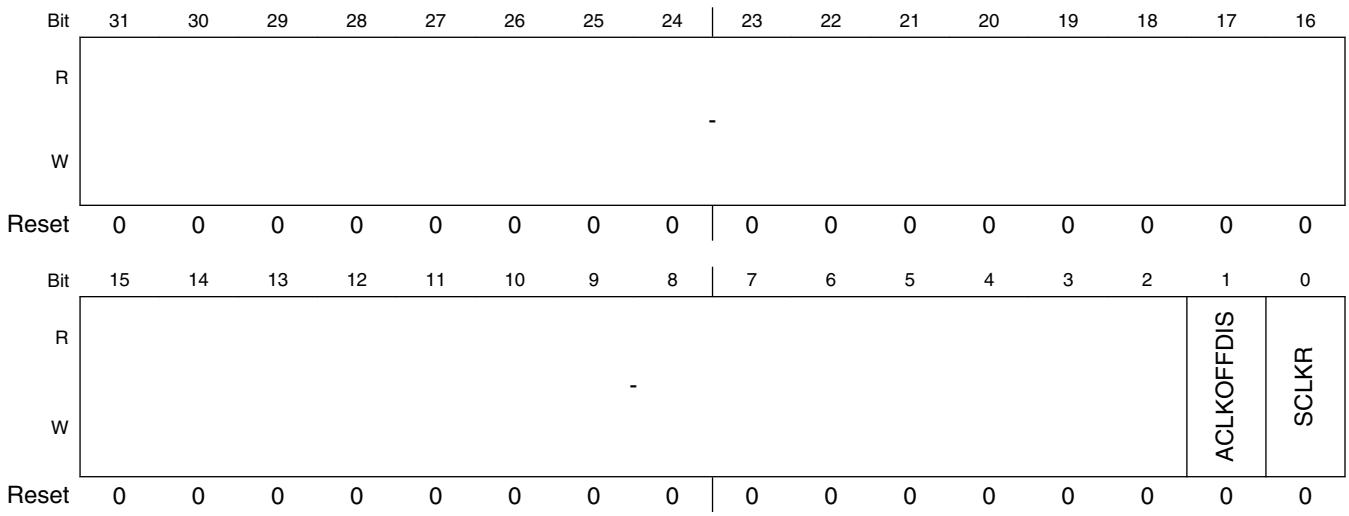
SJC_SSR field descriptions (continued)

Field	Description
	10 Response was entered and is incorrect 11 Response is correct
10–9 SJM	SJC Secure mode Secure JTAG mode, as set by external fuses. 00 No debug (#1) 01 Secure JTAG (#2) 10 Reserved 11 JTAG enabled (#3)
8 FT	Fuse type Fuse type bit - e-fuse or laser fuse 0 E-fuse technology 1 Laser fuse technology
7 -	This field is reserved. Reserved
6 -	This field is reserved. Reserved
5 EBG	External boot granted External boot enabled, requested and granted 1 granted 0 not granted
4 EBF	External Boot fuse Status of the external boot disable fuse 0 (intact) - external boot is allowed 1 (burned) - external boot is disabled
3 SWE	SW enable SW JTAG enable status 1 enabled 0 disabled
2 SWF	Software JTAG enable fuse Status of the no SW disable JTAG fuse 0 (intact) - SW enable possible 1 (intact) - no SW enable possible
1 KTA	Kill Trace is active 1 active 0 not active
0 KTF	Kill Trace Enable fuse value 0 (intact) - kill trace is never active 1 (burned) - kill trace functionality enabled

56.9.7 General Purpose Clocks Control Register (SJC_GPCCR)

This register is used to configure clock related modes in SOC, see System Configuration chapter for more information. Those bits are directly connected to JTAG outputs. Bit 0 of GPCCR controls SDMA clocks invocation. When out of reset, the SDMA is in sleep mode with no SDMA clock running. Unlike events, debug requests does not wake SDMA if it is in sleep mode. The debug request is recognized by the SDMA only when it exits sleep mode upon reception of an event. To be able to enter debug mode even if no event is triggered, the SDMA clock on bit needs to be set prior to sending the debug request (clear at reset).

Address: 0h base + 7h offset = 7h



SJC_GPCCR field descriptions

Field	Description
31–2 -	Reserved
1 ACLKOFFDIS	Disable/prevent ARM platform clock/power shutdown
0 SCLKR	SDMA Clock ON Register - This bit forces the clock on of the SDMA

Chapter 57
Secure Non-Volatile Storage (SNVS)

57.1 SNVS overview

The low-power (battery-backed) section incorporates a secure real time counter, a monotonic counter, and a general purpose register. This portion of the block is powered by a battery that maintains the state of the SNVS_LP registers when the chip is powered off.

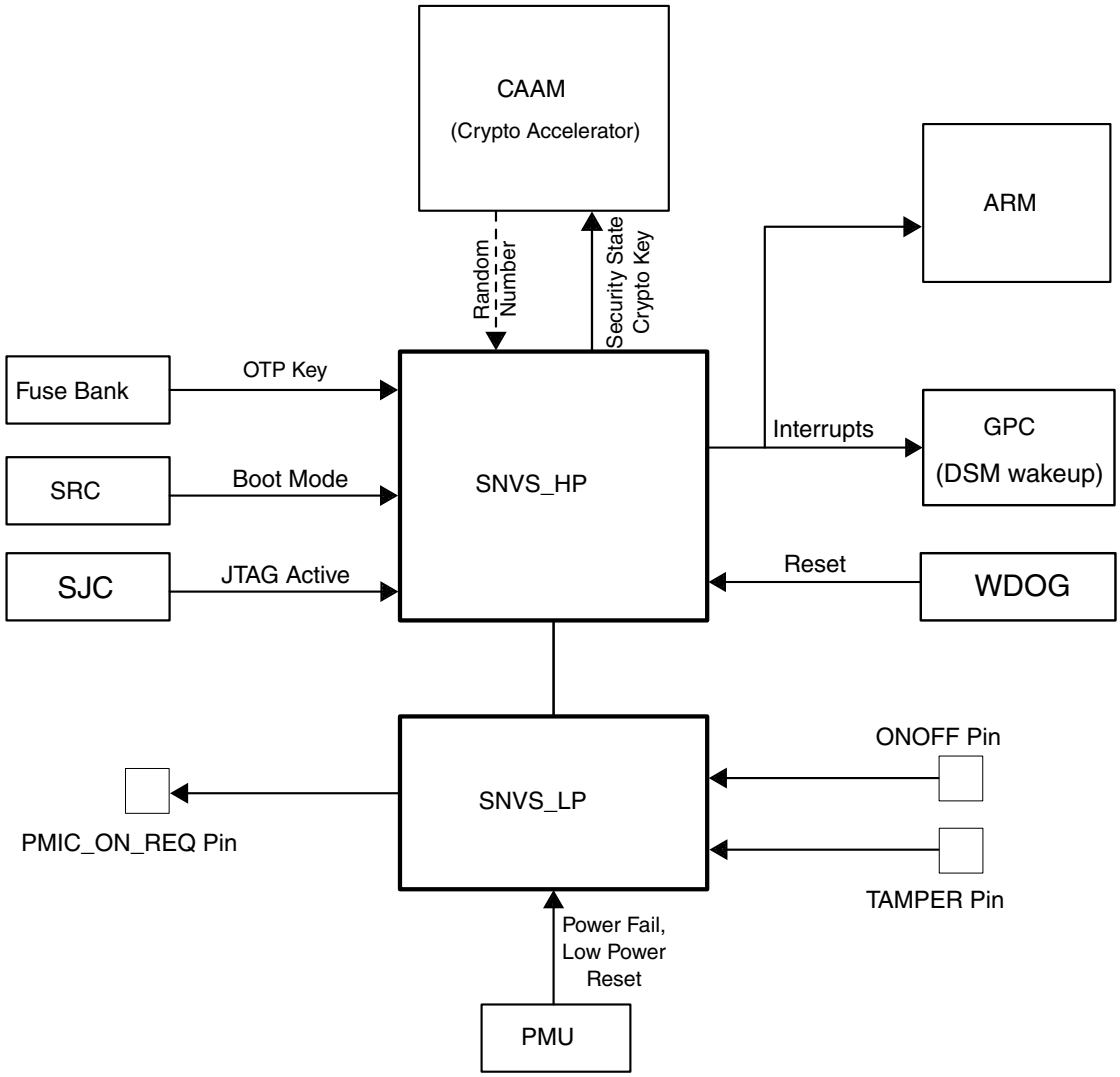


Figure 57-1. Example SNVS Connectivity

NOTE

For the security features of SNVS, see the *Security Reference Manual for i.MX 6Dual, 6Quad, 6Solo, and 6DualLite Families of Application Processors (IMX6DQ6SDLSRM)* or the *Applications Processor Security Reference Manual for i.MX 6SoloLite (IMX6SLSRM)*.

57.1.1 SNVS features

The following table summarizes the features:

Table 57-1. SNVS feature summary

Feature	What it does
Real time counter (RTC)	• The counter is driven by a dedicated clock, which is off when the system power is down • Programmable time alarm interrupt • Periodic interrupt can be generated with different frequencies
Monotonic counter	• The monotonic counter state is nonvolatile. • The counter can only increment. • The counter is a non-rollover counter • The counter value is invalidated in case of security violation.
General-purpose register	• The general-purpose register state is nonvolatile.
Register access protection	• Privileged software access policy • Registers can be programmed only when the system security monitor is in functional state. • Some registers/values can only be programmable once per boot cycle.

57.1.2 Modes of operation

The SNVS operates in either the system power-down or system power-up mode of operation.

During system power-down, SNVS_HP is powered-down. SNVS_LP is powered from the backup power supply and is electrically isolated from the rest of the chip. In this mode, SNVS_LP retains the state of its registers .

During system power-up, SNVS_HP and SNVS_LP are both powered-up and all SNVS functions are operational.

57.2 External Signals

The table found here describes the external signals of SNVS.

Table 57-2. SNVS External Signals

Signal	Description	Pad	Mode	Direction
SNVS_PMIC_ON_REQ	Wake-up signal	PMIC_ON_REQ	Not multiplexed	O
SNVS_TAMPER	Tamper signal	TAMPER	Not multiplexed	I
SNVS_VIO_5	Security violation input signal	GPIO_0	ALT7	I
SNVS_VIO_5_CTL	Security violation output indicator signal for SNVS_VIO_5.	GPIO_18	ALT6	O

57.3 Clocks

The table found here describes the clock sources for SNVS.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 57-3. SNVS Clocks

Clock name	Clock Root	Description
hp_ipg_clk	ipg_clk_root	HP peripheral clock
hp_ipg_clk_s	ipg_clk_root	HP peripheral access clock used for clocking the registers on bus R/W accesses
ipg_hp_rtc_clk	ckil_sync_clk_root	HP RTC clock advances the RTC, doesn't have to be synchrounous with any module clock.
lp_ipg_clk	ipg_clk_root	LP peripheral clock
lp_ipg_clk_s	ipg_clk_root	LP peripheral access clock used for register R/W accesses

57.4 SNVS structure

The SNVS block is divided into two major submodules based on power supply: the high power domain (SNVS_HP) and the low power domain (SNVS_LP). They are powered as follows:

- SNVS_LP - dedicated always-powered-on domain
- SNVS_HP - system (chip) power domain

The following figure illustrates the low power and chip power domains of SNVS.

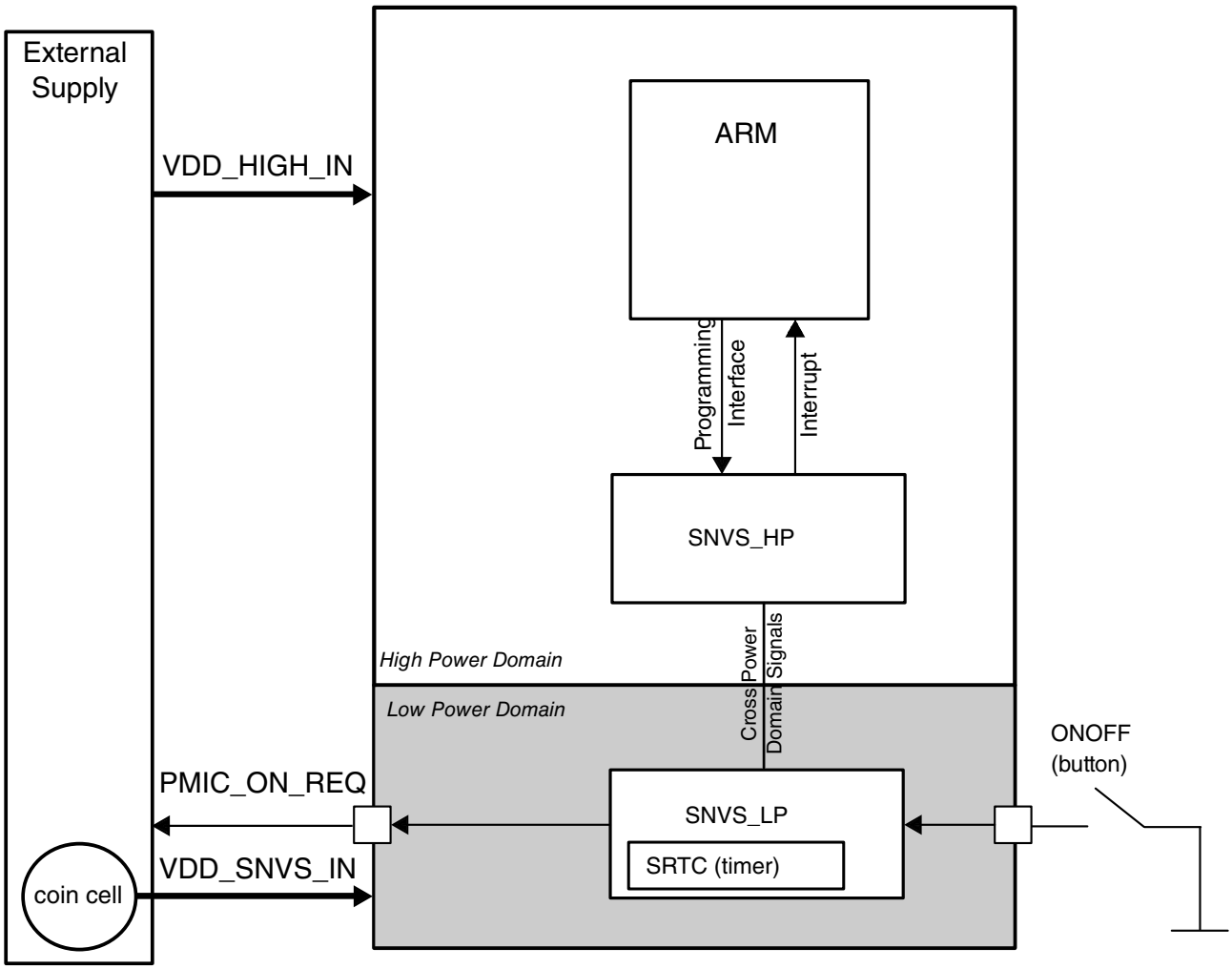


Figure 57-2. SNVS Power Domains

The SNVS_HP section implements all features that enable system communication and provisioning of the SNVS_LP section.

The SNVS_LP section provides hardware that enables secure storage and protection of sensitive data.

57.4.1 SNVS_HP (high power domain)

SNVS_HP is partitioned into the following functional units:

- IP bus interface
- SNVS_LP interface
- Real time counter with alarm
- Control and status registers

SNVS_HP is in the chip's power supply domain and thus receives power along with the rest of the chip. SNVS_HP provides an interface between SNVS_LP and the rest of the system; there is no way to access the SNVS_LP registers except through the SNVS_HP. For access to the SNVS_LP registers, SNVS_HP must be powered up. It uses a register access permission policy to determine whether access to particular registers is permitted.

57.4.2 Non-secure real time counter

SNVS _HP has an autonomous non-secure real time counter. The counter is not active and is reset when the system is powered down. The HP RTC can be used by any application; it has no privileged software access restrictions. The counter can be synchronized with the SNVS_LP SRTC by writing to a specific bit in the SNVS_HP Control Register.

57.4.2.1 Calibrating the time counter

The RTC accuracy may suffer from a drift in the clock, which is used to increment the RTC register. To compensate for this drift, a clock calibration mechanism can adjust the RTC value. It is up to the system processor to decide whether calibration is required or not. If RTC correction is required, enable the mechanism and set the calibration value in the control register. The calibration value is a 5 bit value including the sign bit, which is implemented in 2's complement.

If the calibration mechanism is enabled, the calibration value is added or subtracted from the RTC on a periodic basis, once per 32768 cycles of the RTC clock.

The following table shows the available correction range.

Table 57-4. Time counter calibration settings

Calibration value setting	Correction in counts per 32768 cycles of the counter clock
01111	+15
:	:
00010	+2
00001	+1
00000	0
11111	-1
11110	-2
:	:
10001	-15
10000	-16

57.4.2.2 Time counter alarm

The SNVS _HP non-secure RTC has its own time alarm register. Any application can update this register. The SNVS _HP time alarm can generate interrupts to alert the host processor and can wake up the host processor from one of its low-power modes. Note that this alarm cannot wake up the entire system if it is powered off because this alarm would also be powered off.

57.4.2.3 Periodic interrupt

The SNVS_HP non-secure RTC incorporates a periodic interrupt. The periodic interrupt is generated when a zero-to-one or one-to-zero transition occurs on the selected bit of the RTC. The periodic interrupt source is chosen from 16 bits of the HP RTC according to the PI_FREQ field setting in the HP Control Register. This bit selection also defines the frequency of the periodic interrupt.

The following figure shows the SNVS_HP RTC and its interrupts.

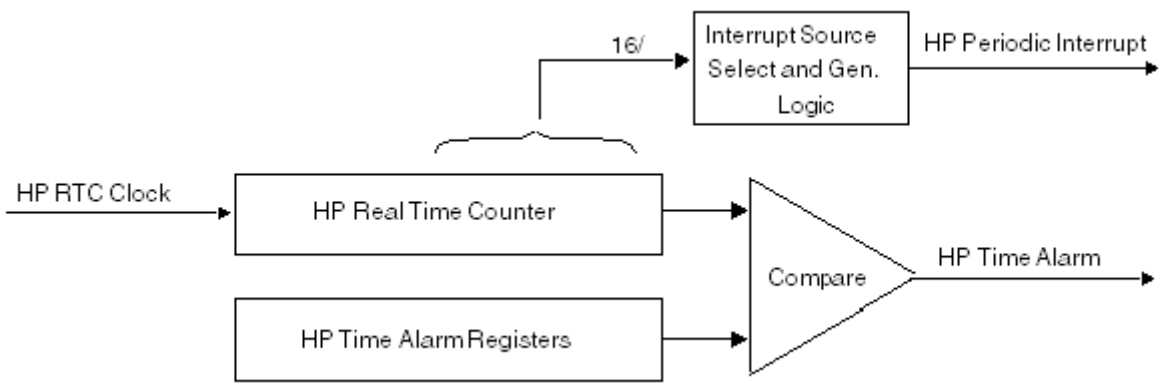


Figure 57-3. SNVS_HP RTC, alarm, and interrupts

57.5 SNVS_LP (low power domain)

SNVS_LP has the following functional units:

- Non-rollover monotonic counter
- General purpose register
- Control and status registers

The SNVS_LP is a data storage subsystem. Its purpose is to store and protect system data, regardless of the main system power state.

SNVS _LP is in the always-powered-up domain, which is a separate power domain with its own power supply.

57.5.1 Behavior during system power down

When the chip power supply domain loses power, SNVS_LP continues to operate normally, and it ignores all inputs from SNVS_HP.

57.5.2 Monotonic counter (MC)

The following figure shows the MC and its rollover security violation.

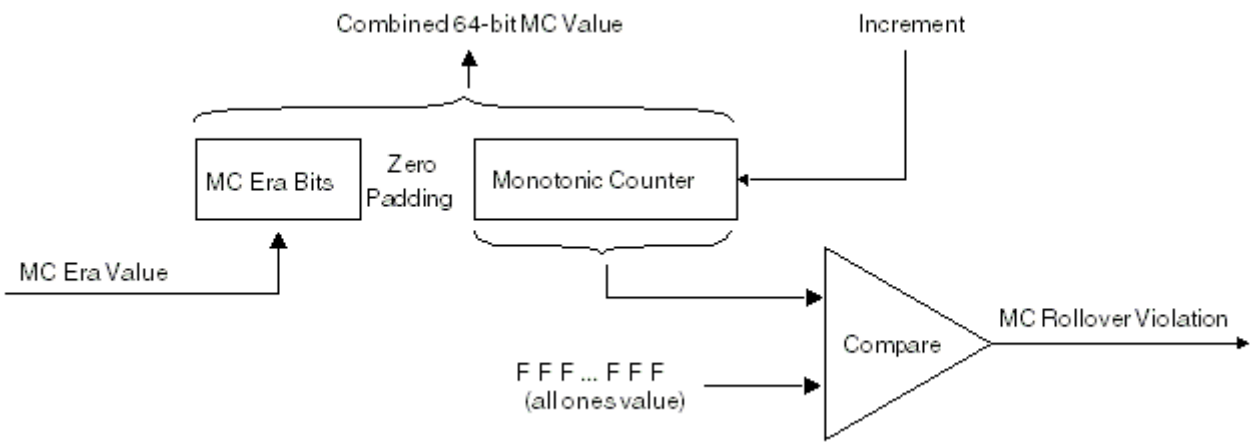


Figure 57-4. SNVS _LP monotonic counter

Some security applications require a monotonic counter (MC) that cannot be exhausted or returned to any previous value during the product's lifetime. Because the MC can never repeat a number, it cannot be reset or cycled back to its starting count. If it reaches its maximum value, it does not rollover. Instead, a monotonic counter rollover indication is generated to the SNVS _LP tamper monitor. This generates an interrupt to the host processor.

The SNVS uses an ERA value derived from the OTP elements as a mechanism for recovery from an MC failure (for example, due to a failure of LP power) where the MC value was compromised or cleared. The ERA value is prepended to the MC to form its

most significant bits. Once any of the ERA value bits are set, the MC can count up from any value, including zero. This guarantees that any future value of the combined monotonic counter will be greater than any of its past values.

57.6 SNVS reset and system power up

This table describes reset actions for SNVS.

Table 57-5. Reset summary

Reset	Source	Characteristics	Internally resets
HP Hard	ipg_hard_async_reset_b	active-low, asynchronous	All SNVS_HPSNVS_LP registers and flops.
LP Power On Reset (POR)	lp_por_b	active-low, asynchronous	All SNVS_LP registers and flops
LP software Reset	software	active-high, synchronous, 1 cycle	All SNVS_LP registers and flops. LP software Reset can be asserted if not disabled.

57.6.1 PMIC Interface

The On/Off logic inside of SNVS_LP allows for connecting directly to a PMIC or other voltage regulator device. The logic takes a button input signal and then outputs a PMIC "ON" Request and a "Power Off" Interrupt. PMIC logic also supports the SNVS_LP tamper logic which will allow waking the system up when a tamper event has happened while in the OFF state. The logic has two different modes of operation (Dumb and Smart mode).

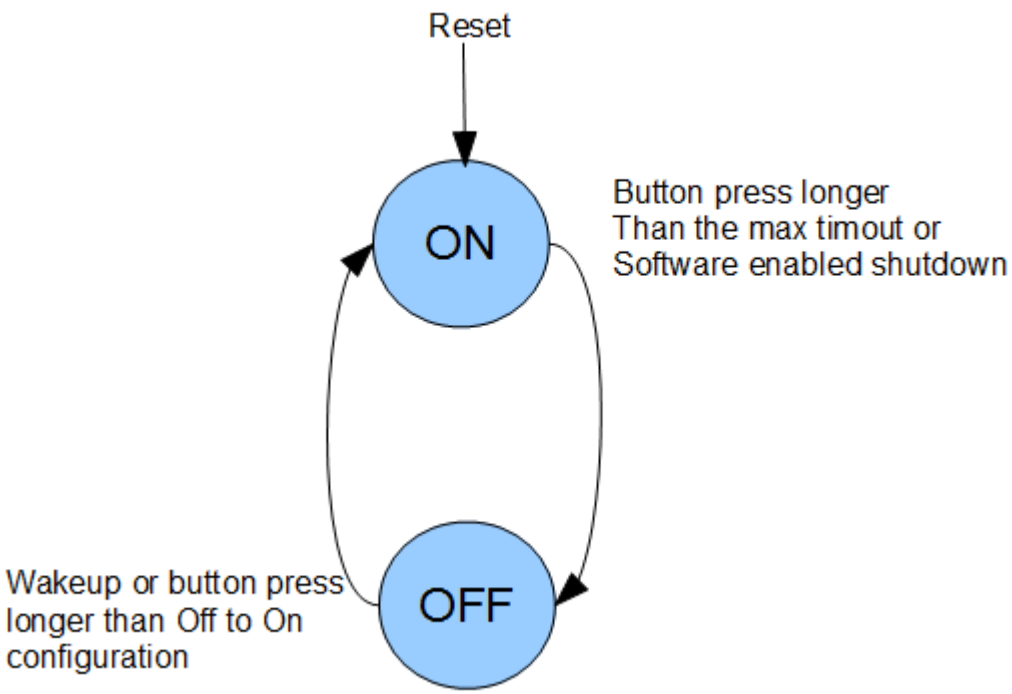
Dumb PMIC Mode:

The dumb pmic mode uses PMIC "ON" Request to issue a level signal for on and off. Dumb pmic mode has many different configuration options which include (debounce, off to on time, and max time out).

- Debounce: The debounce configuration supports 0 msec, 50 msec, 100 msec and 500 msec. The debounce is used to generate the set_pwr_off_irq interrupt. While in the ON state and the button is pressed longer than the debounce time the set_pwr_off_irq is generated.

- Off to On Time: The Off to On configuration supports 0 msec, 50 msec, 100 msec, and 500 msec. This configuration supports the time it takes to request power on after the configured button press time has been reached. Once the button is pressed longer than the configuration time, the state machine will transition from the OFF to the ON state.
- Max Timeout: The max timeout configuration supports 5 secs, 10 secs, 15 secs and disable. This configuration supports the time it takes to request power down after the button has been pressed for the defined time.

The dumb PMIC mode uses a 2 state state machine, as shown below. The output of the pmic_en_b is generated by the state of the state machine.



Smart PMIC Mode:

The smart PMIC mode is meant to connect to another PMIC. The PMIC "ON" Request signal issues a pulse instead of a level signal. The only configuration option available for this mode is the Debounce configuration that is used for the "Power Off" Interrupt.

57.7 SNVS interrupts and alarms

SNVS provides the following interrupt and alarm lines:

- Functional interrupt (active low)
- Real-time clock period interrupt
- Power off (button) interrupt

The following table summarizes all SNVS interrupts and alarm sources.

Table 57-6. Interrupts and alarms summary

Interrupt	Source	Default configuration[1]	Configuration options
SNVS functional interrupt	RTC time alarm	Disable	Enable/Disable
	RTC periodic interrupt	Disable	Enable/Disable
SNVS power off (button) interrupt	BTN input signal	50msec debounce	Debounce time

1. Default behavior refers to the setting after LP/HP reset.

57.8 Programming Guidelines

This section provides initialization and application information for the SNVS module.

57.8.1 RTC control bits setting

All SNVS registers are programmed from the register bus. Therefore, any changes are synchronized with the IP clock. Several registers can also change synchronously with the RTC clock after they are programmed. To avoid IP clock and RTC clock synchronization issues, these values can only be programmed when the corresponding function is disabled. The following table presents the list of these values with the control bit setting required for programming.

Table 57-7. RTC synchronized values list

Function	Value/register	Control bit setting
HP section		
HP Real Time Counter	HPRTCMR and HPRTCLR Registers	RTC_EN = 0 - HPRTCMR/HPRTCLR can be programmed RTC_EN = 1 - HPRTCMR/HPRTCLR cannot be programmed
HP Time Alarm	HPTAMR and HPTALR Registers	HPTA_EN = 0 - HPTAMR/HPTALR can be programmed HPTA_EN = 1 - HPTAMR/HPTALR cannot be programmed
HP Time Calibration Value	HPCALB_VAL Value	HPCALB_EN = 0 - HPCALB_VAL can be programmed HPCALB_EN = 1 - HPCALB_VAL cannot be programmed

Use the following step to program synchronized values:

1. Check the enable bit value. If set, clear it.
2. Verify that the enable bit is cleared.

   There are two reasons to verify the enable bit's setting:

   - Enable bit clearing does not happen immediately; it takes three IPclock cycles and two RTC clock cycles to change the enable bit's value.
   - If the enable bit is locked for programming, it cannot be cleared.
3. Program the desired value.
4. Set the enable bit; it takes three IP clock cycles and two RTC clock cycles for the bit to set.

NOTE

Incrementing the value programmed into RTC registers by two compensates for the two RTC clock cycle delay that is required to enable the counter.

57.8.2 RTC value read

There are two scenarios when software can read corrupted values from the RTC (HPRTCMR and HPRTCLR) registers:

- The RTC counters are incremented by the slow 32 kHz clock, which is asynchronous to the system clock. The counter value is synchronized to the system clock before software reads that. The synchronization register may capture the counter value in the middle of the counter update. In this case, it is not guaranteed that all bits are properly sampled by the synchronization register; the value read by software can be wrong.
- The RTC value is longer than the single bus read transaction of 32-bits. Therefore, software reads two registers, each holding a portion of the counter value. After reading one of these registers but before reading the second register, both registers may update their values. In this case, the value combined by software will be incorrect.

To avoid these issues, it is strongly recommended that software perform two consecutive reads of the RTC value:

- If two consecutive reads are similar, the value is correct.
- If two consecutive reads are different, perform two more reads.

The worst case scenario may require three sessions of two consecutive reads.

57.8.3 General initialization guidelines

Complete the following steps in order to properly initialize the module:

1. Enable interrupts in SNVScontrol and configuration registers.
2. Program SNVS general functions/configurations.
3. User Specific: Set lock bits.

NOTE

57.9 SNVS Memory Map/Register Definition

This section contains detailed register descriptions for the SNVS registers. Each description includes a standard register diagram and register table. The register table provides detailed descriptions of the register bit and field functions, in bit order.

SNVS registers consist of two types:

- Privileged read/write accessible
- Non-privileged read/write accessible

Privileged read/write accessible registers can only be accessed for read/write by privileged software. Unauthorized write accesses are ignored, and unauthorized read accesses return zero. Non-privileged software can access privileged access registers when the non-privileged software access enable bit is set in the SNVS_HP Command Register.

- Non-Secure
- Trusted
- Secure

Non-privileged read/write accessible registers are read/write accessible by any software.

The following table shows the SNVS memory map. The LP register values are set only on LP POR and are unaffected by System (HP) POR. The HP registers are set only on System POR and are unaffected by LP POR.

NOTE

For more information on security-related bitfields, see the *Security Reference Manual for i.MX 6Dual, 6Quad, 6Solo, and 6DualLite Families of Application Processors*

(IMX6DQ6SDLSRM) or the *Applications Processor Security Reference Manual for i.MX 6SoloLite (IMX6SLSRM).*

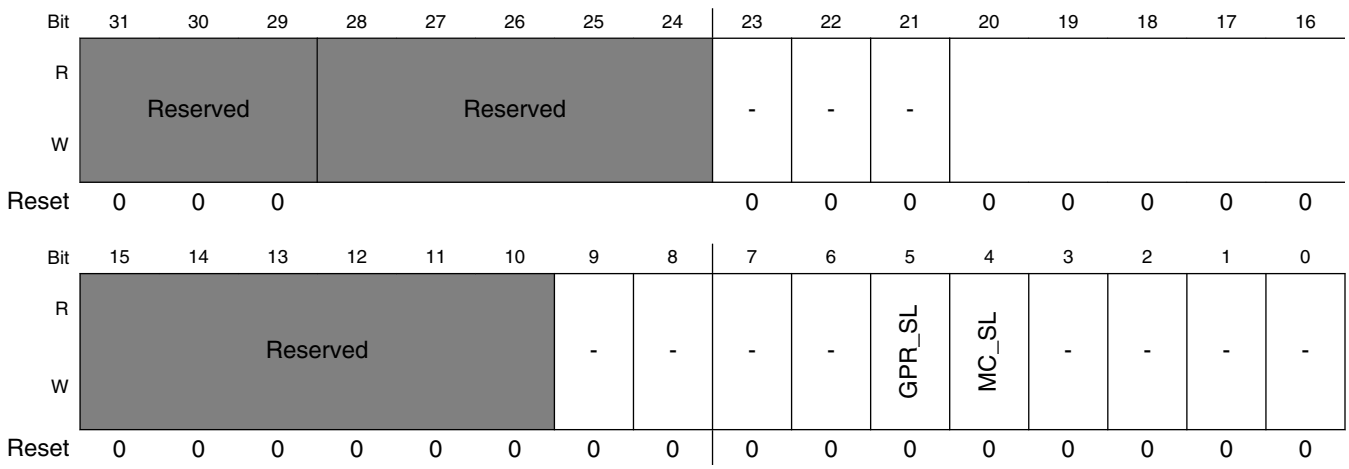
SNVS memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20C_C000	SNVS_HP Lock Register (SNVS_HPLR)	32	R/W	0000_0000h	57.9.1/4983
20C_C004	SNVS_HP Command Register (SNVS_HPCOMR)	32	R/W	0000_0000h	57.9.2/4985
20C_C008	SNVS_HP Control Register (SNVS_HPCR)	32	R/W	0000_0000h	57.9.3/4987
20C_C014	SNVS_HP Status Register (SNVS_HPSR)	32	R/W	8000_0000h	57.9.4/4990
20C_C024	SNVS_HP Real Time Counter MSB Register (SNVS_HPRTCMR)	32	R/W	0000_0000h	57.9.5/4992
20C_C028	SNVS_HP Real Time Counter LSB Register (SNVS_HPRTCLR)	32	R/W	0000_0000h	57.9.6/4993
20C_C02C	SNVS_HP Time Alarm MSB Register (SNVS_HPTAMR)	32	R/W	0000_0000h	57.9.7/4993
20C_C030	SNVS_HP Time Alarm LSB Register (SNVS_HPTALR)	32	R/W	0000_0000h	57.9.8/4994
20C_C034	SNVS_LP Lock Register (SNVS_LPLR)	32	R/W	0000_0000h	57.9.9/4994
20C_C038	SNVS_LP Control Register (SNVS_LPCR)	32	R/W	0000_0000h	57.9.10/ 4996
20C_C04C	SNVS_LP Status Register (SNVS_LPSR)	32	R/W	0000_0008h	57.9.11/ 4999
20C_C05C	SNVS_LP Secure Monotonic Counter MSB Register (SNVS_LPSMCMR)	32	R/W	0000_0000h	57.9.12/ 5001
20C_C060	SNVS_LP Secure Monotonic Counter LSB Register (SNVS_LPSMCLR)	32	R/W	0000_0000h	57.9.13/ 5002
20C_C068	SNVS_LP General Purpose Register (SNVS_LPGPR)	32	R/W	0000_0000h	57.9.14/ 5002
20C_CBF8	SNVS_HP Version ID Register 1 (SNVS_HPVIDR1)	32	R	003E_0100h	57.9.15/ 5003
20C_CBFC	SNVS_HP Version ID Register 2 (SNVS_HPVIDR2)	32	R	0000_0000h	57.9.16/ 5003

57.9.1 SNVS_HP Lock Register (SNVS_HPLR)

The SNVS_HP Lock Register contains lock bits for the SNVS registers. This is a privileged write register.

Address: 20C_C000h base + 0h offset = 20C_C000h



SNVS_HPLR field descriptions

Field	Description
31–29 -	This field is reserved.
23–19 -	This field is reserved.
18 -	This field is reserved.
17 -	This field is reserved.
16 -	This field is reserved.
15–10 -	This field is reserved.
9 -	This field is reserved.
8 -	This field is reserved.
7 -	This field is reserved.
6 -	This field is reserved.
5 GPR_SL	General Purpose Register Soft Lock

Table continues on the next page...

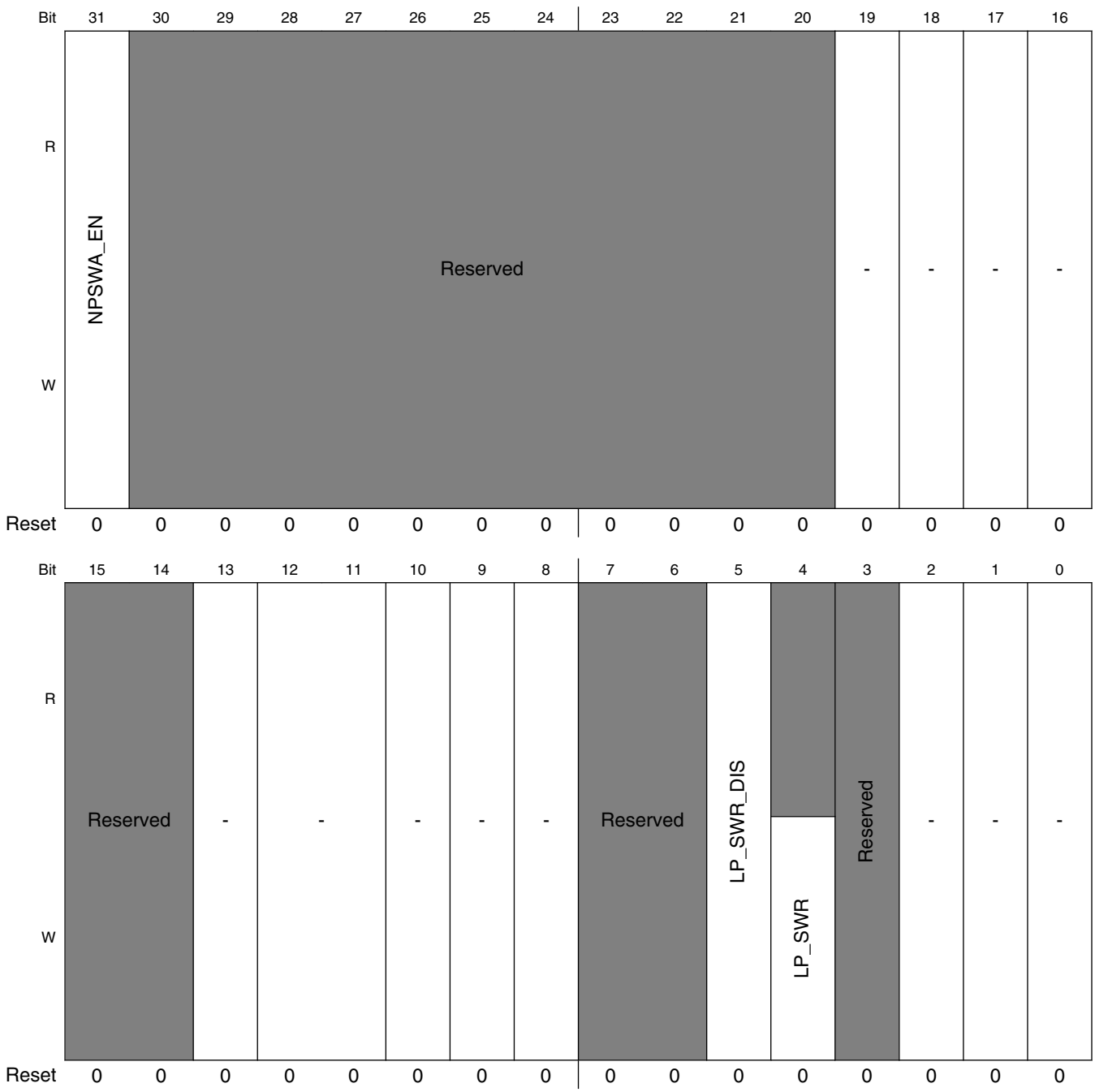
SNVS_HPLR field descriptions (continued)

Field	Description
	When set, prevents any writes to the GPR. Once set, this bit can only be reset by the system reset. 0 Write access is allowed 1 Write access is not allowed
4 MC_SL	Monotonic Counter Soft Lock When set, prevents any writes (increments) to the MC Registers and MC_ENV bit. Once set, this bit can only be reset by the system reset. 0 Write access (increment) is allowed 1 Write access (increment) is not allowed
3 -	This field is reserved.
2 -	This field is reserved.
1 -	This field is reserved.
0 -	This field is reserved.

57.9.2 SNVS_HP Command Register (SNVS_HPCOMR)

The SNVS_HP Command Register contains the command, configuration, and control bits for the SNVS block. This is a privileged write register.

Address: 20C_C000h base + 4h offset = 20C_C004h



SNVS_HPCOMR field descriptions

Field	Description
31 NPSWA_EN	Non-Privileged Software Access Enable When set, allows non-privileged software to access all SNVS registers, including those that are privileged software read/write access only. 0 Only privileged software can access privileged registers 1 Any software can access privileged registers
30–20 -	This field is reserved.
19 -	This field is reserved.
18 -	This field is reserved.
17 -	This field is reserved.
16 -	This field is reserved.
15–14 -	This field is reserved.
13 -	This field is reserved.
12–11 -	This field is reserved.
10 -	This field is reserved.
9 -	This field is reserved.
8 -	This field is reserved.
7–6 -	This field is reserved.
5 LP_SWR_DIS	LP Software Reset Disable When set, disables the LP software reset. Once set, this bit can only be reset by the system reset. 0 LP software reset is enabled 1 LP software reset is disabled
4 LP_SWR	LP Software Reset When set, it resets the SNVS_LP section. This bit cannot be set when the LP_SWR_DIS bit is set. This self-clearing bit is always read as zero. 0 No Action 1 Reset LP section
3 -	This field is reserved.
2 -	This field is reserved.
1 -	This field is reserved.

Table continues on the next page...

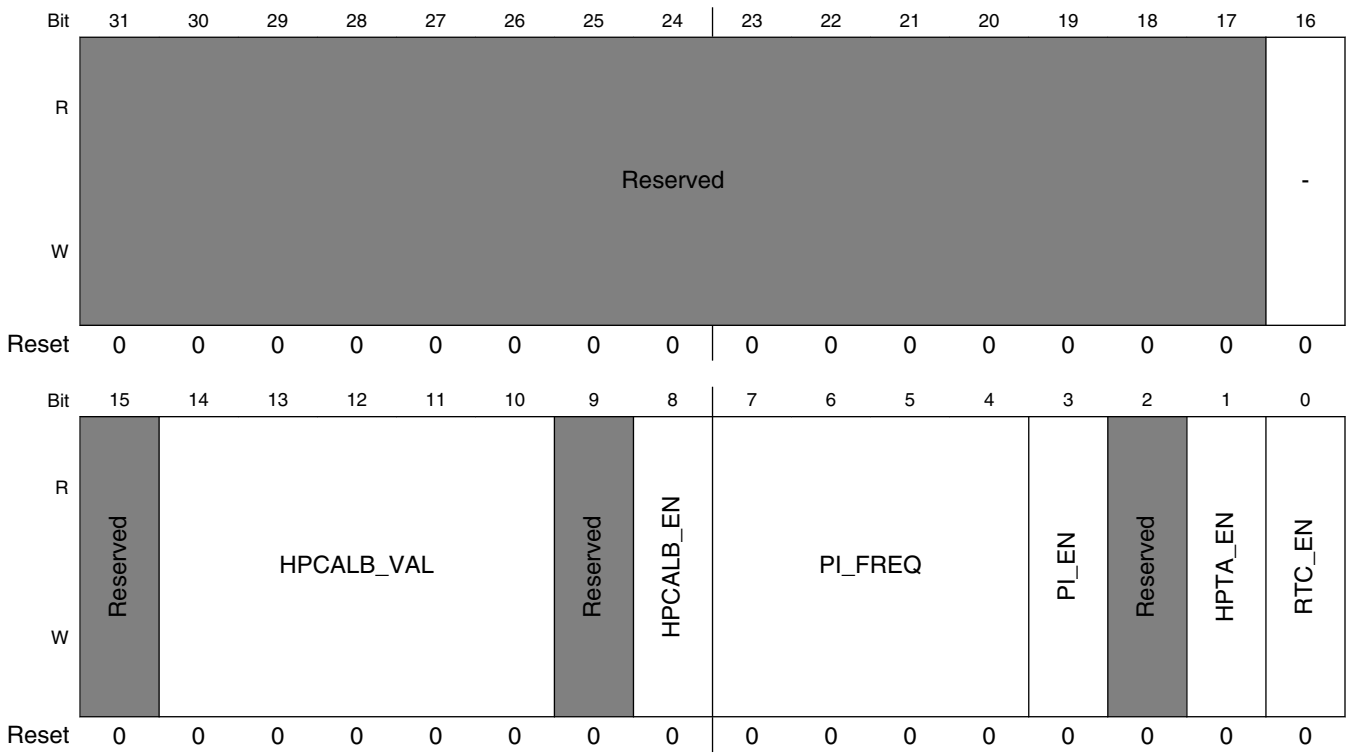
SNVS_HPCOMR field descriptions (continued)

Field	Description
0 -	This field is reserved.

57.9.3 SNVS_HP Control Register (SNVS_HPCR)

The SNVS_HP Control Register contains various control bits of the HP section of SNVS .

Address: 20C_C000h base + 8h offset = 20C_C008h



SNVS_HPCR field descriptions

Field	Description
31–17 -	This field is reserved.
16 -	This field is reserved.
15 -	This field is reserved.

Table continues on the next page...

SNVS_HPCR field descriptions (continued)

Field	Description
14–10 HPCALB_VAL	HP Calibration Value Defines signed calibration value for the HP Real Time Counter. This field can be programmed only when RTC Calibration is disabled (HPCALB_EN is not set). This is a 5-bit 2's complement value, hence the allowable calibration values are in the range from -16 to +15 counts per 32768 ticks of the counter. 00000 +0 counts per each 32768 ticks of the counter 00001 +1 counts per each 32768 ticks of the counter 00010 +2 counts per each 32768 ticks of the counter 01111 +15 counts per each 32768 ticks of the counter 10000 -16 counts per each 32768 ticks of the counter 10001 -15 counts per each 32768 ticks of the counter 11110 -2 counts per each 32768 ticks of the counter 11111 -1 counts per each 32768 ticks of the counter
9 -	This field is reserved.
8 HPCALB_EN	HP Real Time Counter Calibration Enabled Indicates that the time calibration mechanism is enabled. 0 HP Timer calibration disabled 1 HP Timer calibration enabled
7–4 PI_FREQ	Periodic Interrupt Frequency Defines frequency of the periodic interrupt. The interrupt is generated when a zero-to-one or one-to-zero transition occurs on the selected bit of the HP Real Time Counter and Real Time Counter and Periodic Interrupt are both enabled (RTC_EN and PI_EN are set). It is recommended to program this field when Periodic Interrupt is disabled (PI_EN is not set). The possible frequencies are: 0000 - bit 0 of the RTC is selected as a source of the periodic interrupt 0001 - bit 1 of the RTC is selected as a source of the periodic interrupt 0010 - bit 2 of the RTC is selected as a source of the periodic interrupt 0011 - bit 3 of the RTC is selected as a source of the periodic interrupt 0100 - bit 4 of the RTC is selected as a source of the periodic interrupt 0101 - bit 5 of the RTC is selected as a source of the periodic interrupt 0110 - bit 6 of the RTC is selected as a source of the periodic interrupt 0111 - bit 7 of the RTC is selected as a source of the periodic interrupt 1000 - bit 8 of the RTC is selected as a source of the periodic interrupt 1001 - bit 9 of the RTC is selected as a source of the periodic interrupt 1010 - bit 10 of the RTC is selected as a source of the periodic interrupt 1011 - bit 11 of the RTC is selected as a source of the periodic interrupt 1100 - bit 12 of the RTC is selected as a source of the periodic interrupt 1101 - bit 13 of the RTC is selected as a source of the periodic interrupt 1110 - bit 14 of the RTC is selected as a source of the periodic interrupt 1111 - bit 15 of the RTC is selected as a source of the periodic interrupt
3 PI_EN	HP Periodic Interrupt Enable The periodic interrupt can be generated only if the HP Real Time Counter is enabled. 0 HP Periodic Interrupt is disabled 1 HP Periodic Interrupt is enabled

Table continues on the next page...

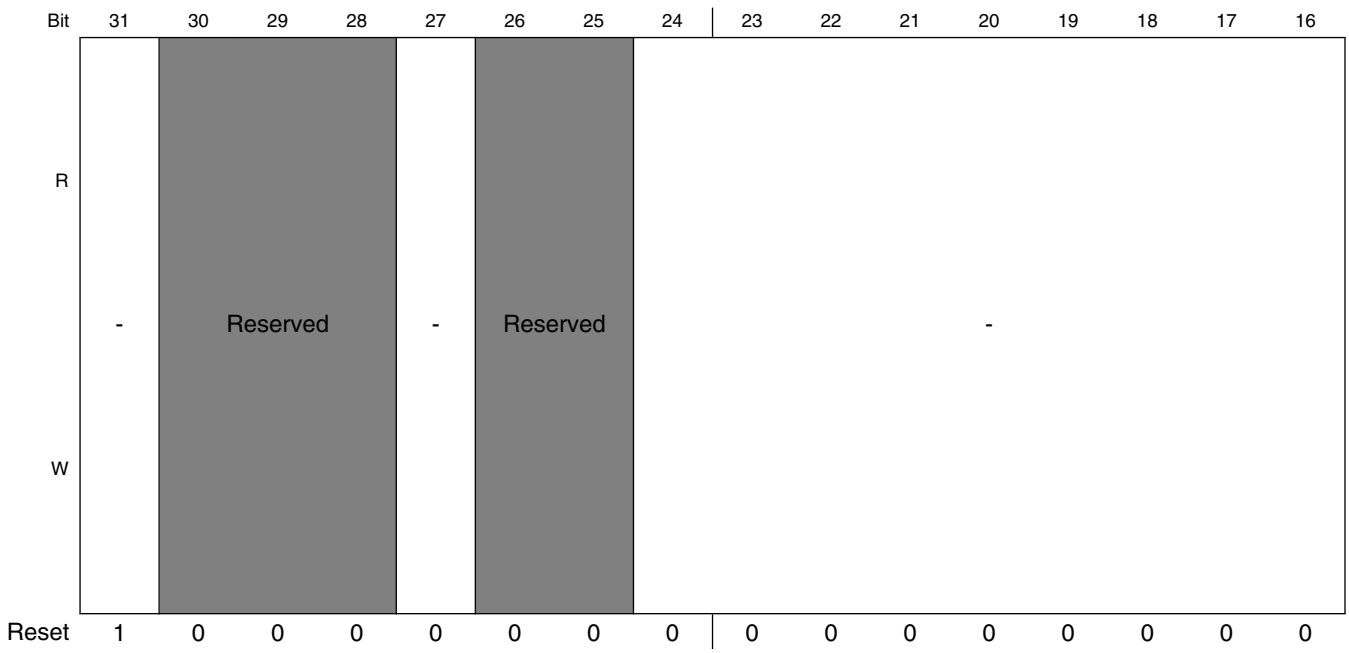
SNVS_HPCR field descriptions (continued)

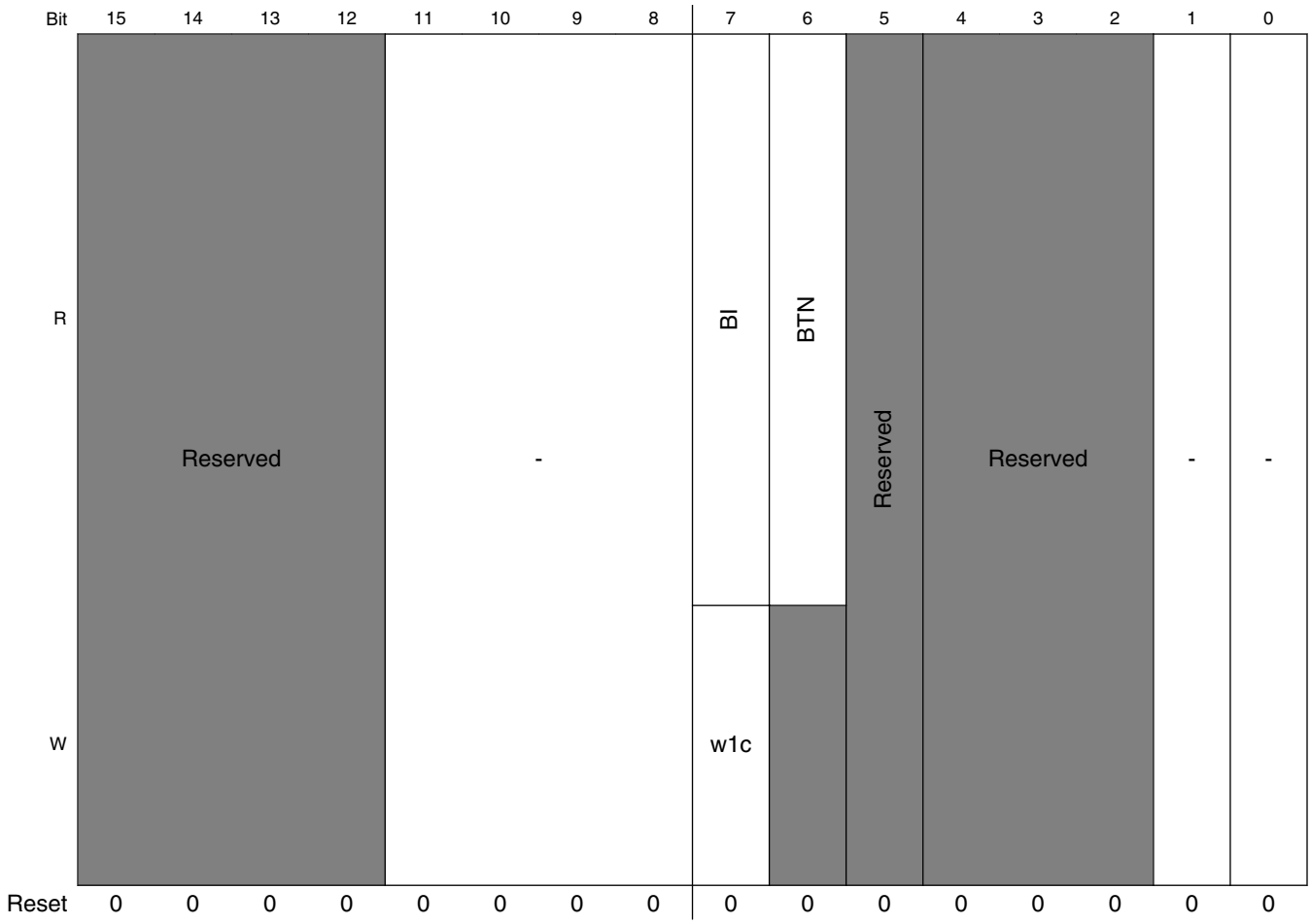
Field	Description
2 -	This field is reserved.
1 HPTA_EN	HP Time Alarm Enable When set, the time alarm interrupt is generated if the value in the HP Time Alarm Registers is equal to the value of the HP Real Time Counter. 0 HP Time Alarm Interrupt is disabled 1 HP Time Alarm Interrupt is enabled
0 RTC_EN	HP Real Time Counter Enable 0 RTC is disabled 1 RTC is enabled

57.9.4 SNVS_HP Status Register (SNVS_HPSR)

The HP Status Register reflects the internal state of the SNVS.

Address: 20C_C000h base + 14h offset = 20C_C014h





SNVS_HPSR field descriptions

Field	Description
31 -	This field is reserved.
30–28 -	This field is reserved.
27 -	This field is reserved.
26–25 -	This field is reserved.
24–16 -	This field is reserved.
15–12 -	This field is reserved.
11–8 -	This field is reserved.
7 BI	Button Interrupt. Signal ipi_snvs_btn_int_b was asserted.
6 BTN	Value of the BTN input. This is the external button used for PMIC control.

Table continues on the next page...

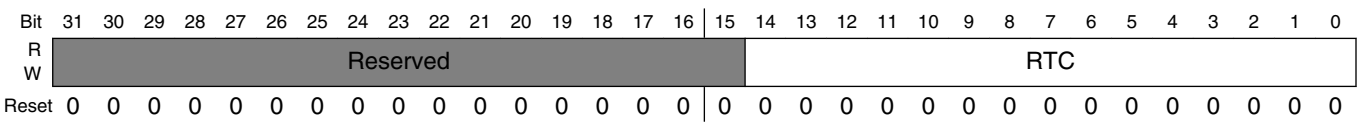
SNVS_HPSR field descriptions (continued)

Field	Description
	0: BTN not pressed 1: BTN pressed
5 -	This field is reserved.
4–2 -	This field is reserved.
1 -	This field is reserved.
0 -	This field is reserved.

57.9.5 SNVS_HP Real Time Counter MSB Register (SNVS_HPRTCMR)

The SNVS_HP Real Time Counter MSB register contains the most significant bits of the HP Real Time Counter.

Address: 20C_C000h base + 24h offset = 20C_C024h



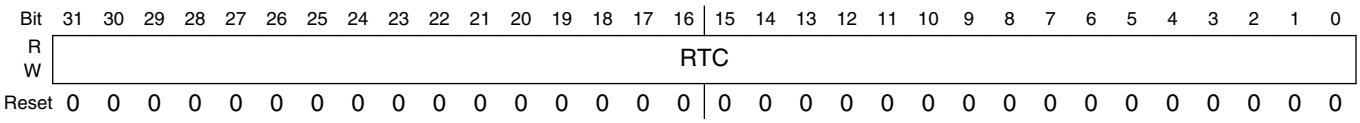
SNVS_HPRTCMR field descriptions

Field	Description
31–15 -	This field is reserved. Reserved
RTC	HP Real Time Counter Most significant 15 bits. This register can be programmed only when RTC is not active (RTC_EN bit is not set).

57.9.6 SNVS_HP Real Time Counter LSB Register (SNVS_HPRTCLR)

The SNVS_HP Real Time Counter LSB register contains the 32 least significant bits of the HP real time counter.

Address: 20C_C000h base + 28h offset = 20C_C028h



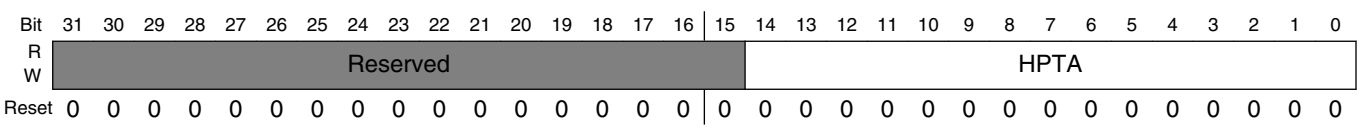
SNVS_HPRTCLR field descriptions

Field	Description
RTC	HP Real Time Counter Least significant 32 bits. This register can be programmed only when RTC is not active (RTC_EN bit is not set).

57.9.7 SNVS_HP Time Alarm MSB Register (SNVS_HPTAMR)

The SNVS_HP Time Alarm MSB register contains the most significant bits of the SNVS_HP Time Alarm value.

Address: 20C_C000h base + 2Ch offset = 20C_C02Ch



SNVS_HPTAMR field descriptions

Field	Description
31–15 -	This field is reserved.
HPTA	HP Time Alarm Most significant 15 bits. This register can be programmed only when HP time alarm is disabled (HPTA_EN bit is not set).

57.9.8 SNVS_HP Time Alarm LSB Register (SNVS_HPTALR)

The SNVS_HP Time Alarm LSB register contains the 32 least significant bits of the SNVS_HP Time Alarm value.

Address: 20C_C000h base + 30h offset = 20C_C030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	HPTA																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SNVS_HPTALR field descriptions

Field	Description
HPTA	HP Time Alarm Least significant bits. This register can be programmed only when HP time alarm is disabled (HPTA_EN bit is not set).

57.9.9 SNVS_LP Lock Register (SNVS_LPLR)

The SNVS_LP Lock Register contains lock bits for the SNVS_LP registers.

Address: 20C_C000h base + 34h offset = 20C_C034h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved			Reserved												
Reset	0	0	0						0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved						-	-	-	-	GPR_HL	MC_HL	-	-	-	-
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

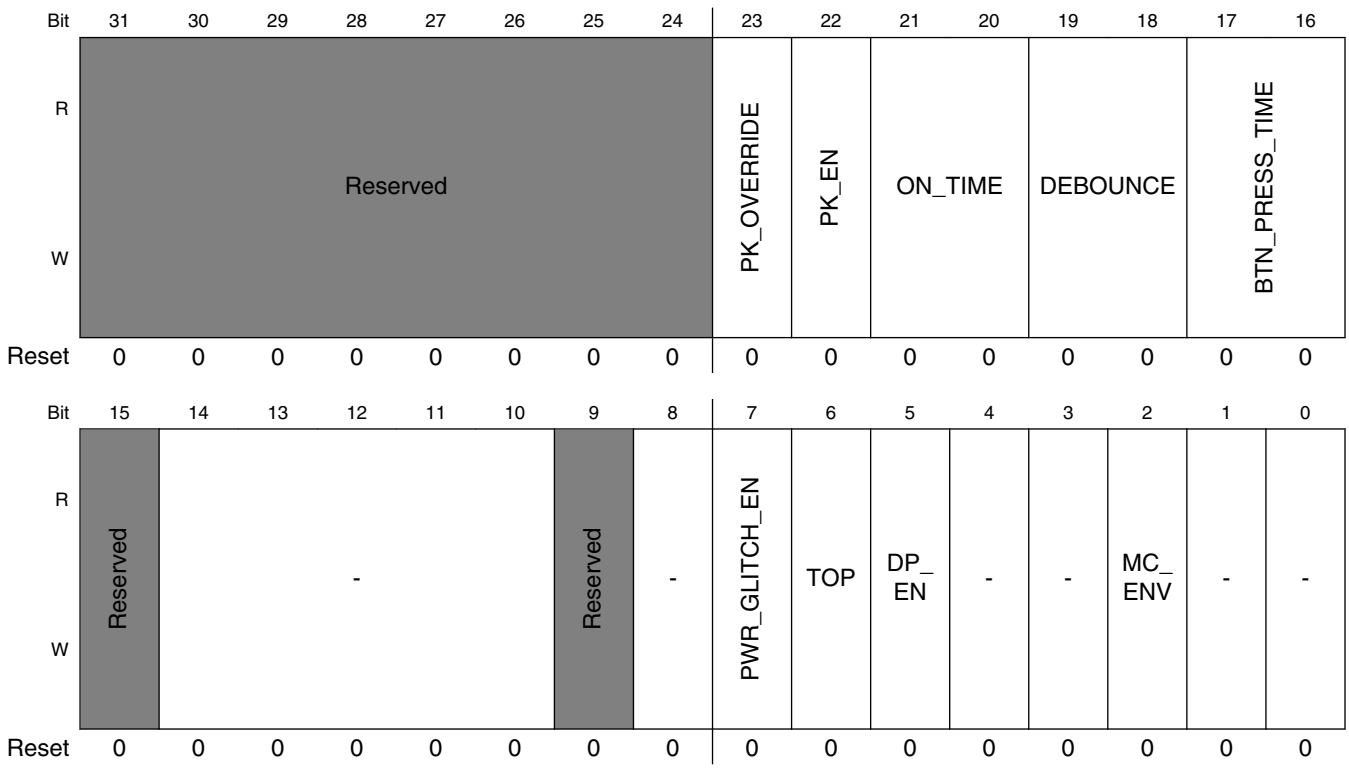
SNVS_LPLR field descriptions

Field	Description
31–29 -	This field is reserved.
23–10 -	This field is reserved.
9 -	This field is reserved.
8 -	This field is reserved.
7 -	This field is reserved.
6 -	This field is reserved.
5 GPR_HL	General Purpose Register Hard Lock When set, prevents any writes to the GPR. Once set, this bit can only be reset by the LP POR. 0 Write access is allowed. 1 Write access is not allowed.
4 MC_HL	Monotonic Counter Hard Lock When set, prevents any writes (increments) to the MC Registers and MC_ENV bit. Once set, this bit can only be reset by the LP POR. 0 Write access (increment) is allowed. 1 Write access (increment) is not allowed.
3 -	This field is reserved.
2 -	This field is reserved.
1 -	This field is reserved.
0 -	This field is reserved.

57.9.10 SNVS_LP Control Register (SNVS_LPCR)

The SNVS_LP Control Register contains various control bits of the LP section of SNVS.

Address: 20C_C000h base + 38h offset = 20C_C038h



SNVS_LPCR field descriptions

Field	Description
31–24 -	This field is reserved.
23 PK_OVERRIDE	PMIC On Request Override. The value written to PK_OVERRIDE will be asserted on output signal snvs_lp_pk_override. That signal is used to override the IOMUX control for the PMIC I/O pad.
22 PK_EN	PMIC On Request Enable. The value written to PK_EN will be asserted on output signal snvs_lp_pk_en. That signal is used to turn off the pullup/pulldown circuitry in the PMIC I/O pad.
21–20 ON_TIME	The ON_TIME field is used to configure the period of time after BTN is asserted before pmic_en_b is asserted to turn on the SoCpower. 00: 500msec off->on transition time 01: 50msec off->on transition time 10: 100msec off->on transition time 11: 0msec off->on transition time

Table continues on the next page...

SNVS_LPCR field descriptions (continued)

Field	Description
19–18 DEBOUNCE	This field configures the amount of debounce time for the BTN input signal. 00: 50msec debounce 01: 100msec debounce 10: 500msec debounce 11: 0msec debounce
17–16 BTN_PRESS_TIME	Button press time out values for PMIC Logic. 00 : 5 secs 01 : 10 secs 10 : 15 secs 11 : long press disabled (pmic_en_b will not be asserted regardlessof how long BTN is asserted)
15 -	This field is reserved.
14–10 -	This field is reserved.
9 -	This field is reserved.
8 -	This field is reserved.
7 PWR_GLITCH_EN	By default the detection of a power glitch does not cause the pmic_en_b signal to be asserted. Setting the Power Glitch Enable bit to 1 enables the power glitch event for the PMIC. 0 - disabled 1 - enabled
6 TOP	Turn off System Power Asserting this bit causes a signal to be sent to the Power Management IC to turn off the system power. This bit will clear once power is off. This bit is only valid when the Dumb PMIC is enabled. 0 Leave system power on. 1 Turn off system power.
5 DP_EN	Dumb PMIC Enabled When set, software can control the system power. When cleared, the system requires a Smart PMIC to automatically turn power off. 0 Smart PMIC enabled. 1 Dumb PMIC enabled.
4 -	This field is reserved.
3 -	This field is reserved.
2 MC_ENV	Monotonic Counter Enable and Valid When set, the MC can be incremented (by write transaction to the LPSMCMR or LPSMCLR). This bit cannot be changed once MC_SL or MC_HL bit is set. 0 MC is disabled or invalid. 1 MC is enabled and valid.

Table continues on the next page...

SNVS_LPCR field descriptions (continued)

Field	Description
1 -	This field is reserved.
0 -	This field is reserved.

57.9.11 SNVS_LP Status Register (SNVS_LPSR)

The SNVS_LP Status Register reflects the internal state and behavior of the SNVS_LP.

Address: 20C_C000h base + 4Ch offset = 20C_C04Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-	-	Reserved									-	Reserved	SPO	EO	-
W														w1c	w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved					-	-	-	-	-	-	-	-	MCR	-	-
W														w1c		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

SNVS_LPSR field descriptions

Field	Description
31 -	This field is reserved.
30 -	This field is reserved.
29–21 -	This field is reserved.
20 -	This field is reserved.
19 -	This field is reserved.
18 SPO	Set Power Off The SPO bit is set when the set_pwr_off_irq interrupt is triggered, which happens when software writes a 1 to the TOP bit in the LPCR or when the power button is pressed longer than the configured debounce time. Writing to the SPO bit will clear the set_pwr_off_irq interrupt. 0 Emergency Off was not detected. 1 Emergency Off was detected..
17 EO	Emergency Off This bit is set when a power off is requested. 0 Emergency off was not detected. 1 Emergency off was detected.
16 -	This field is reserved.
15–11 -	This field is reserved.
10 -	This field is reserved.
9 -	This field is reserved.
8 -	This field is reserved.
7 -	This field is reserved.
6 -	This field is reserved.
5 -	This field is reserved.
4 -	This field is reserved.
3 -	This field is reserved.
2 MCR	Monotonic Counter Rollover. 0 MC has not reached its maximum value. 1 MC has reached its maximum value.

Table continues on the next page...

SNVS_LPSR field descriptions (continued)

Field	Description
1 -	This field is reserved.
0 -	This field is reserved.

57.9.12 SNVS_LP Secure Monotonic Counter MSB Register (SNVS_LPSMCMR)

The SNVS_LP Secure Monotonic Counter MSB Register contains the monotonic counter era bits and the most significant 16 bits of the monotonic counter. The monotonic counter is incremented by one if there is a write command to the LPSMCMR or LPSMCLR register.

Address: 20C_C000h base + 5Ch offset = 20C_C05Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MC_ERA_BITS																MON_COUNTER															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

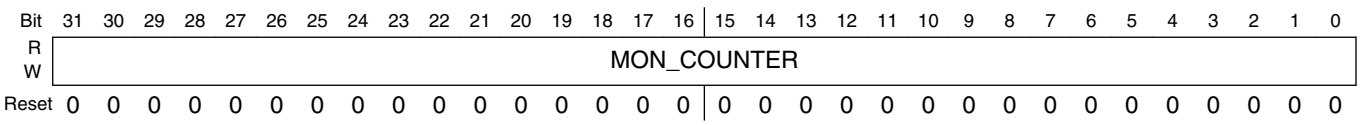
SNVS_LPSMCMR field descriptions

Field	Description
31–16 MC_ERA_BITS	Monotonic Counter Era Bits These bits are inputs to the module and typically connect to fuses.
MON_COUNTER	Monotonic Counter Most Significant 16 Bits The MC is incremented by one when: • A write transaction to the LPSMCMR or LPSMCLR register is detected. • The MC_ENV bit is set. • MC_SL and MC_HL bits are not set.

57.9.13 SNVS_LP Secure Monotonic Counter LSB Register (SNVS_LPSMCLR)

The SNVS_LP Secure Monotonic Counter LSB Register contains the 32 least significant bits of the monotonic counter. The MC is incremented by one if there is a write command to the LPSMCMR or LPSMCLR register.

Address: 20C_C000h base + 60h offset = 20C_C060h



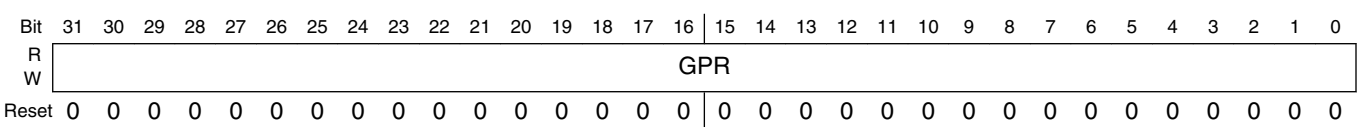
SNVS_LPSMCLR field descriptions

Field	Description
MON_COUNTER	Monotonic Counter bits The MC is incremented by one when: • A write transaction to the LPSMCMR or LPSMCLR Register is detected. • The MC_ENV bit is set. • MC_SL and MC_HL bits are not set.

57.9.14 SNVS_LP General Purpose Register (SNVS_LPGPR)

The SNVS_LP General Purpose Register is a read/write register located in the low power domain, which can be used by any application for retaining data during an SoC power-down mode.

Address: 20C_C000h base + 68h offset = 20C_C068h



SNVS_LPGPR field descriptions

Field	Description
GPR	General Purpose Register When GPR_SL or GPR_HL bit is set, the register cannot be programmed.

57.9.15 SNVS_HP Version ID Register 1 (SNVS_HPVIDR1)

The SNVS_HP Version ID Register 1 is a read-only register that contains the current version of the SNVS . The version consists of a module ID, a major version number, and a minor version number.

Address: 20C_C000h base + BF8h offset = 20C_CBF8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IP_ID																MAJOR_REV								MINOR_REV							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

SNVS_HPVIDR1 field descriptions

Field	Description
31–16 IP_ID	SNVS block ID
15–8 MAJOR_REV	SNVS block major version number
MINOR_REV	SNVS block minor version number

57.9.16 SNVS_HP Version ID Register 2 (SNVS_HPVIDR2)

The SNVS_HP Version ID Register 2 is a read-only register that indicates the current version of the SNVS. Version ID register 2 consists of the following fields: integration options, ECO revision, and configuration options.

Address: 20C_C000h base + BFCh offset = 20C_CBFCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IP_ERA								INTG_OPT								ECO_REV								CONFIG_OPT							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SNVS_HPVIDR2 field descriptions

Field	Description
31–24 IP_ERA	Era of the IP design 00h - Era 1 or 2

Table continues on the next page...

SNVS_HPVIDR2 field descriptions (continued)

Field	Description
	03h - Era 3 04h - Era 4 05h - Era 5
23–16 INTG_OPT	SNVS Integration Option
15–8 ECO_REV	SNVS ECO Revision
CONFIG_OPT	SNVS Configuration Option

Chapter 58
Shared Peripheral Bus Arbiter (SPBA)

58.1 Overview

The Shared Peripheral Bus Arbiter (SPBA) is a three-to-one IP Bus interface arbiter. Three masters arbitrate for shared peripheral access through the SPBA.

The SPBA has three primary functions:

- The IP Bus Line switches a master to one peripheral
- The Masters arbiter arbitrates between the three masters to solve concurrent access or restricted access to peripherals
- The Control Registers and Ownership Control includes a set of registers which are reachable through software and permit the access scheme to be defined for each peripheral (Resource Ownership and Access Control). It generates signals for the external steering logic of interrupts and DMA signals.

The figure below shows the SPBA block diagram

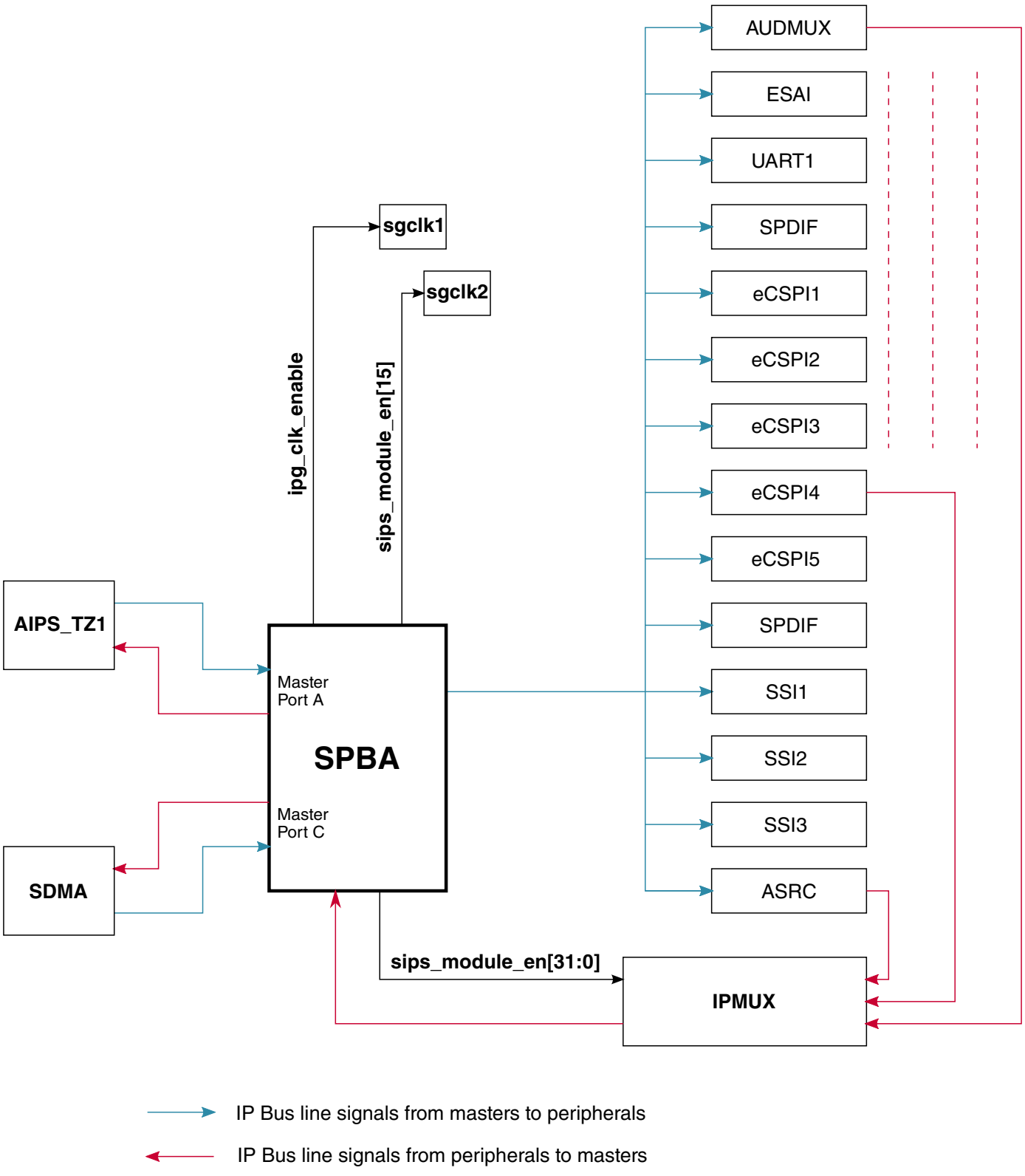


Figure 58-1. i.MX 6Dual/6Quad SPBA connectivity

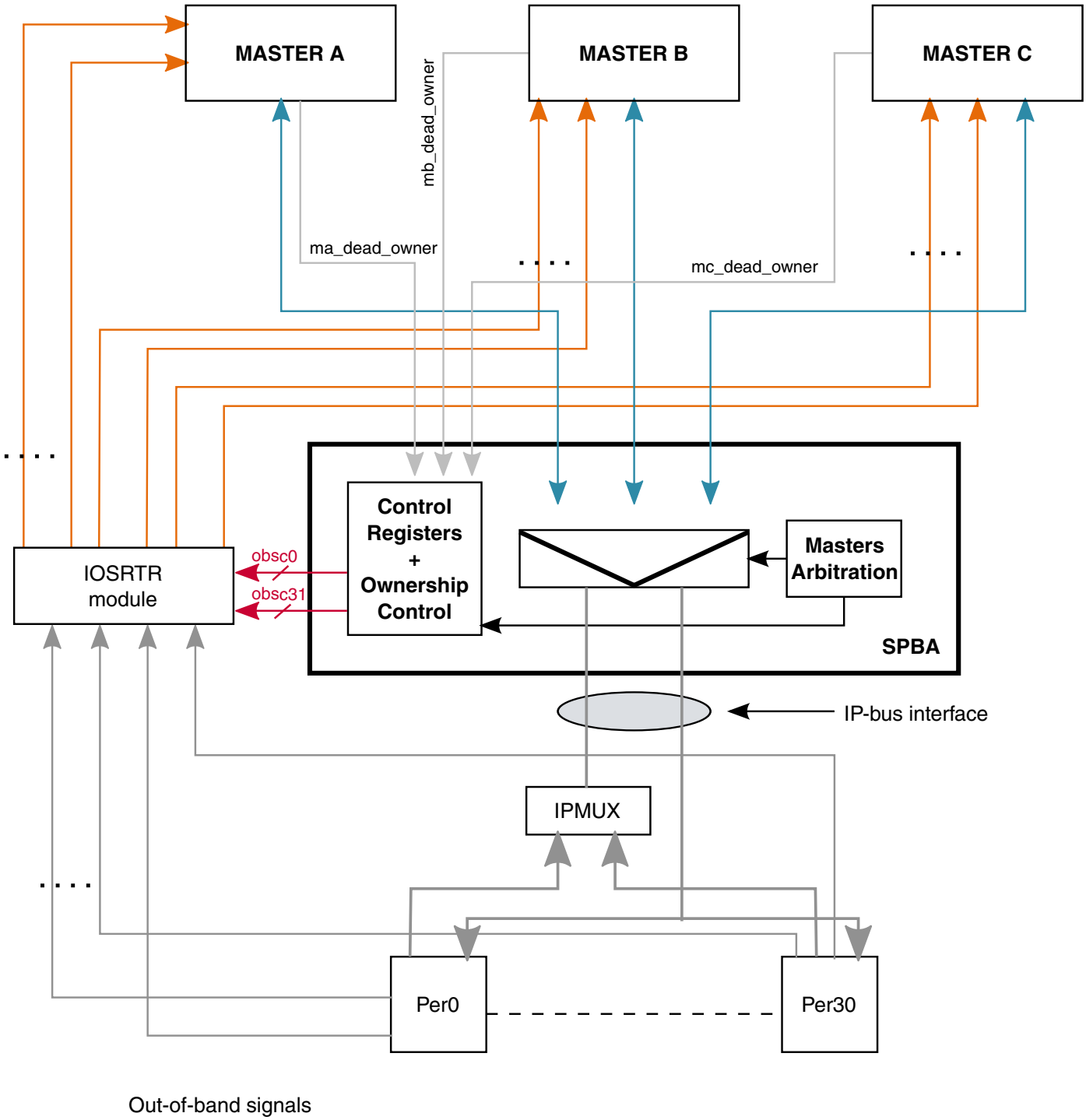


Figure 58-2. SPBA Block Diagram

58.1.1 Features

The SPBA includes the following features:

- Three IP Bus masters arbitration: Master A, B and C
- Support for DMA masters
- 32-bit data
- Supports up to 31 shared peripherals, each consuming 16 kilobytes of address space
- SPBA can be considered the 32nd peripheral, used for resource ownership and access control of the 31 peripherals
- Provides 31 sets of out of band steering control (OBSC) signals to the off-block steering logic
- Operating frequency up to 67 MHz
- Clocks: ipg_clk, ipg_clk_s

58.1.2 Modes of operation

SPBA behavior is transparent when accessing a peripheral, though it has these distinct modes of operation.

Reset/Abort

The SPBA has a hardware reset which initializes all registers, arbitration and peripherals rights registers (PRRs).

An abort signal input is provided allowing each master to abort its current access and release ownership (in case of master reset sequence).

Functional

Once a master request is granted, its IP Bus signals are steered to the requested peripheral.

Standby

No clock needed. The SPBA needs clocks only during access to the PRRs, arbitration, and abort phases. It generates two clock enable signals indicating when the clocks must be provided.

Configuration

During this phase, a master accesses the SPBA PRRs. The SPBA memory-mapped registers are seen as a shared peripheral.

58.2 Clocks

The table found here describes the clock sources for SPBA.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 58-1. SPBA Clocks

Clock name	Clock Root	Description
ipg_clk	ipg_clk_root	Peripheral clock
ipg_clk_s	ipg_clk_root	Peripheral access clock

58.3 Functional description

58.3.1 Masters arbitration

The arbitration mechanism determines which port will control the master port, based on a simple round-robin arbitration scheme.

There are several use cases to consider.

- Only one master request per access. The master is switched to the shared peripheral bus, without arbitration. Figure 58-3 shows the MB request on the global module enable signal, served without wait state.
- If two masters simultaneously access SPBA, the last granted master is held off using the <master>_ips_xfr_wait output signal (default value is high). When the master is granted sips_xfr_wait, shared IP Bus peripheral is connected to <master>_ips_xfr_wait outputs.
- If three masters simultaneously access SPBA, then the last two granted masters are held off using <master>_ips_xfr_wait. Figure 58-4 shows a case in which the last two accesses granted are MA and MB. The requests are used even if they are in the same cycle.
- If after reset, at the first multiple access, no master has been granted, the priority is static: Master A (MA), Master B (MB) and last Master C (MC) port.
- No master request. No master switch to shared peripherals.

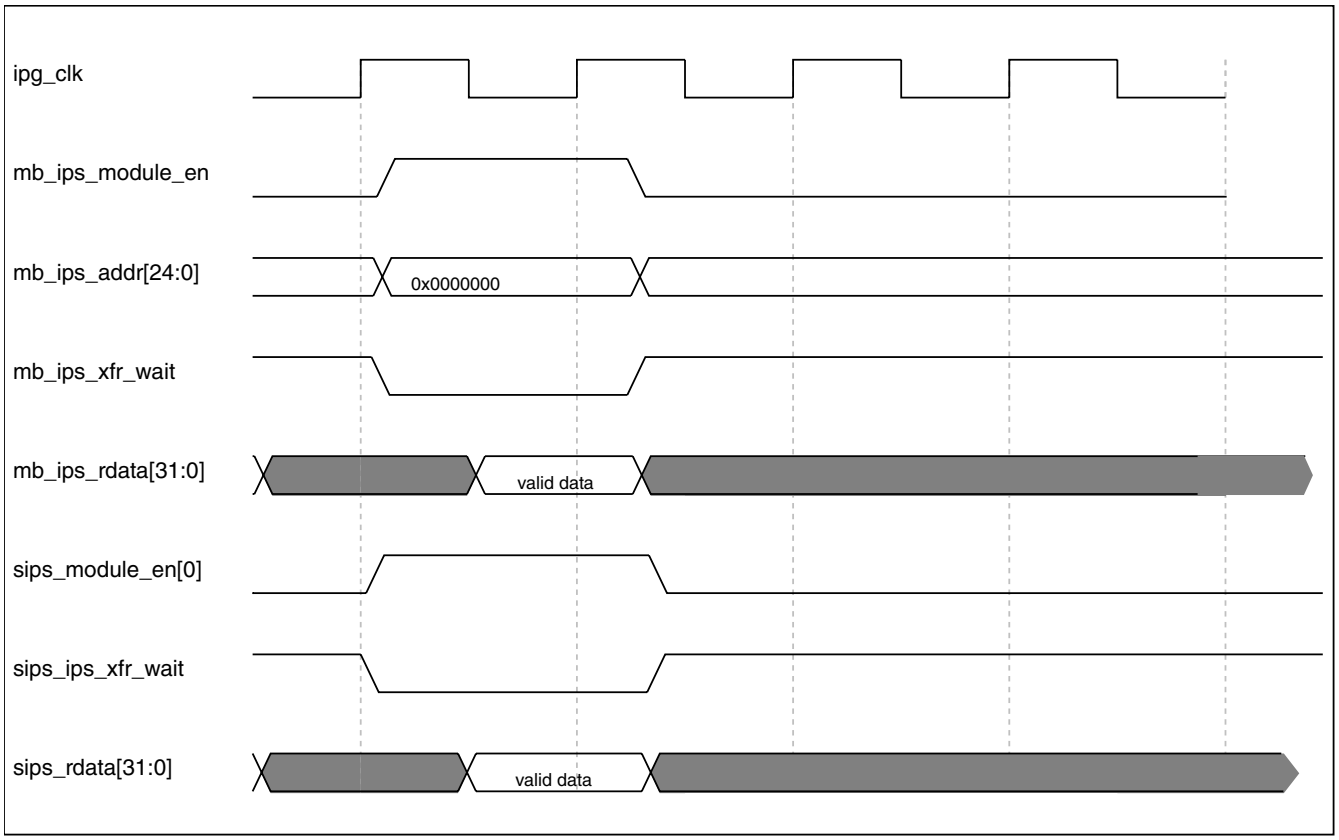


Figure 58-3. Example of one master request, no SPBA arbitration

The following figure assumes MA and MB have been the last two masters granted in the previous transfers (MA then MB).

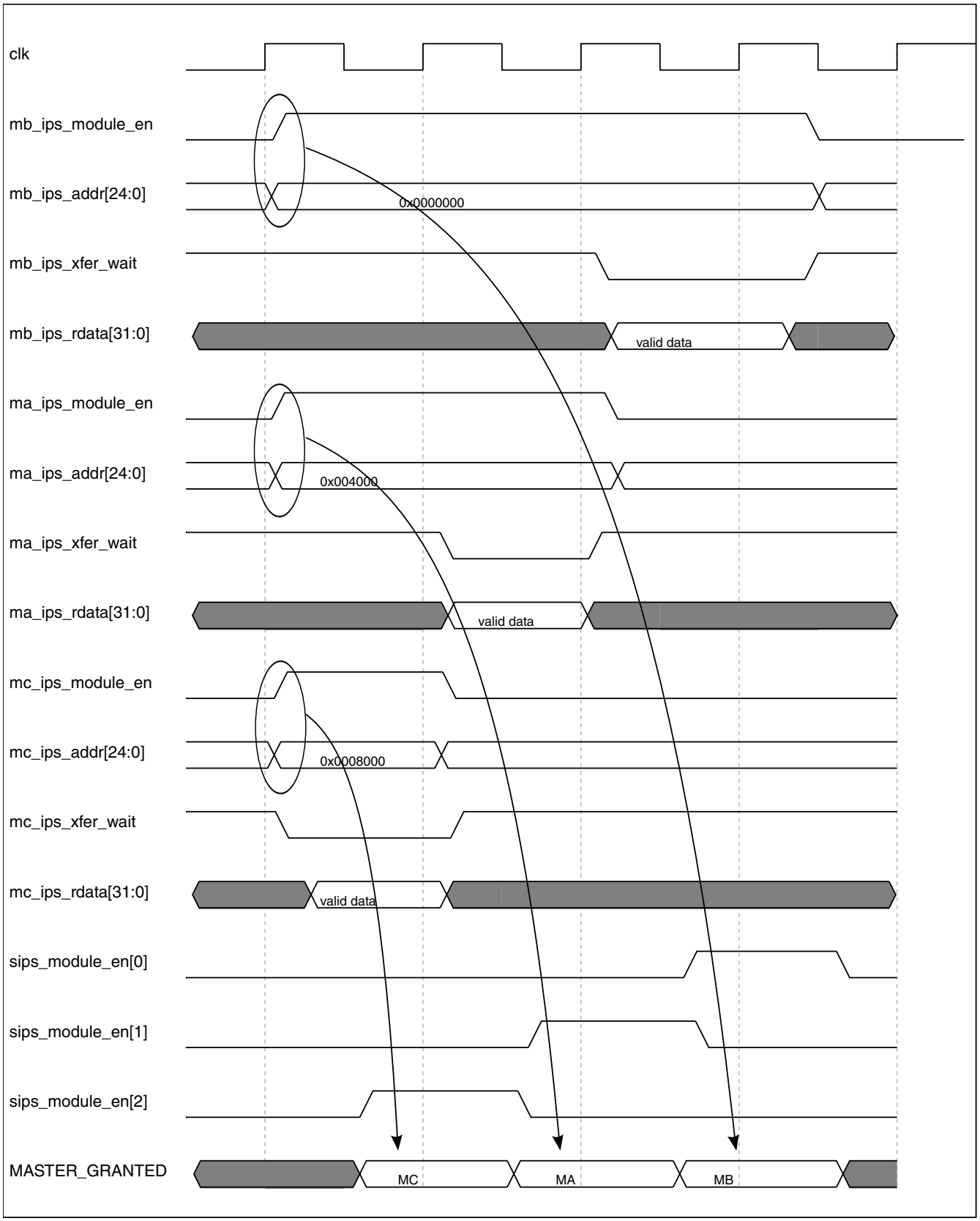


Figure 58-4. Example of three master requests: Masters already granted are "waited";

58.4 Resource ownership control

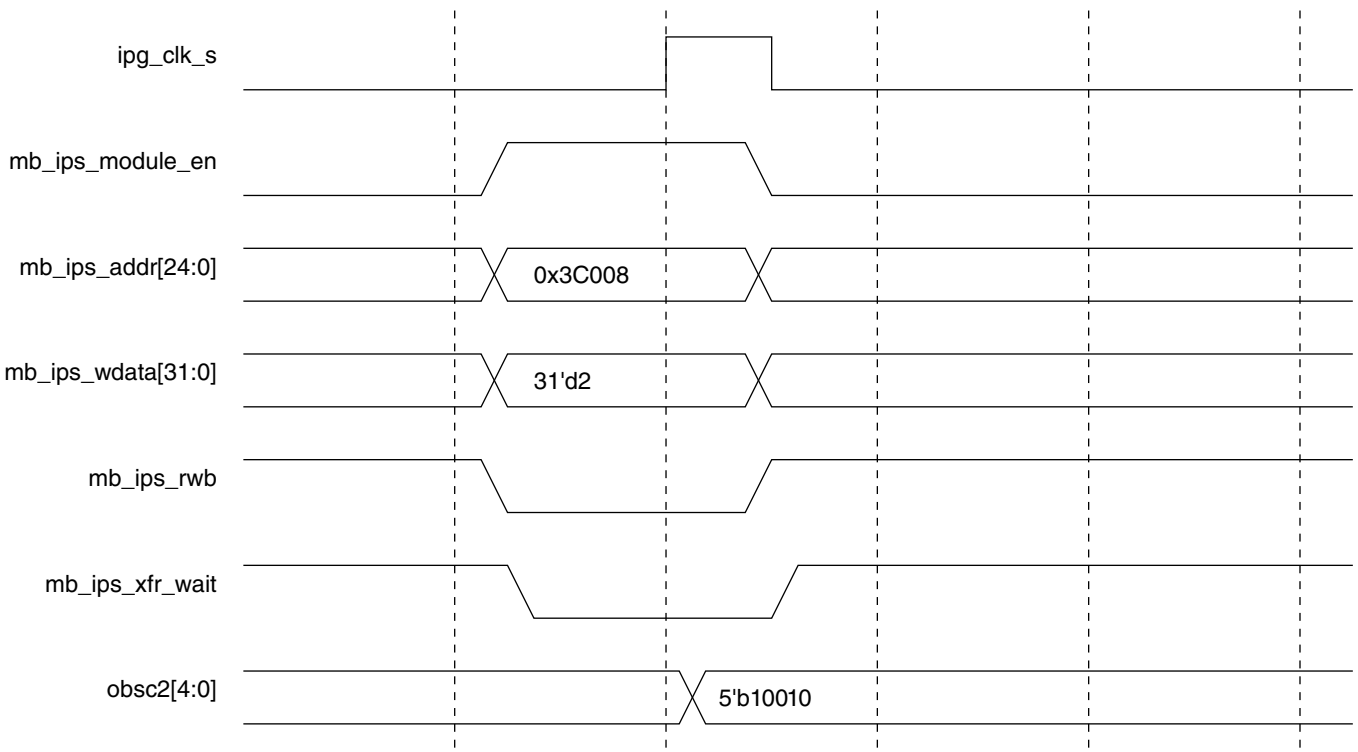
The resource ownership control regulates access to the shared peripherals and determines the steering of out-of-band signals.

58.4.1 Access control

58.4.1.1 Peripheral access

The peripheral access (resource access) of the requesting master is given by the corresponding RAR bit of the Peripheral Right Register. It determines if the master has access privilege to the resource.

Any attempt at access made by a requesting master whose access privilege bit is not set (in the PRR) is terminated with a bus error (**<master>_ips_xfr_err** is asserted by SPBA logic). The master that owns the resource can lock the peripheral for itself and/or grant other masters access to the peripheral by setting the appropriate bit(s) in the RAR field.



Master B is taking ownership of peripheral 2 by writing 3'b010 in the SPBA peripheral 2 right register (rarfield)
This ownership can be checked on obsc2 output as roi2[1:0] = 2'b10 and rar2[2:0] = 3'b010
(obsc[4:0] = {roi2[1], roi2[0], rar2[2], rar2[1], rar2[0]})

Figure 58-5. Example of one master B gaining ownership of peripheral 2

58.4.1.2 Peripheral Right Register access

The ROI bits of the Peripheral Right Register (PRR) determine which master is allowed to make write access to PRR. The identification of the requesting master is compared to the ROI bits of the PRR to determine if the master has ownership of the corresponding register.

Any attempted write access to a PRR already owned by another master will be ignored.

58.4.2 Owner election

When the peripheral is not owned by any master (ROI="00", after coming out of reset for instance), the first master to perform successfully a write to the RAR bits of the PRR is granted ownership of the peripheral and its associated PRR.

After writing to the PRR (RAR bit(s)), the master must read it back to make sure that it was granted ownership. If the RMO field is 2'b11, then the ownership claim is successful. If RMO is 2'b10, another master claimed ownership before this master was able to complete its write. This resolves the case in which two or more masters attempt to write the PRR at the same time; only the first master will be granted ownership. However all masters must read the PRR to determine if this case occurred, and if so, whether they were the first master which was granted ownership.

NOTE

A master that has been granted ownership of the PRR does not automatically have the right access to the peripheral; it must still set its own RAR bits in the PRR to access the peripheral.

58.4.3 Ending ownership

Ownership may be voluntarily ended by the owning master, or automatically upon assertion of a master-specific dead_owner signal.

The former is appropriate for software-controlled yielding of ownership. The latter is appropriate for automatic yielding of ownership when the owner has gone into reset.

When a master is reset, it clears the ROI bits of the PRRs owned by the corresponding master. When the owner is dead (in reset), all peripherals previously owned by that master must be changed to the un-owned state.

NOTE

It is the programmer's responsibility to make sure the peripherals are placed in an appropriate state before ending ownership.

58.4.3.1 Software Controlled Ownership Ending

The ROI bits will be automatically cleared when the master that owns the PRR access right clears (write) the RAR bits (Table 2).

It will then end the ownership of the PRR.

58.4.4 The Un-owned State

During the time when the peripheral is un-owned (i.e the ROI field contains all 0's), all masters have full access to it (RAR bits can then be modified by a master if ROI[1:0] = 2'b0).

In such cases it is necessary for software to ensure any necessary coherency in the resource, there is no hardware protection.

58.5 SPBA Memory Map/Register Definition

The SPBA control registers (Peripheral Right Registers) are mapped as a virtual shared peripheral.

SPBA can support up to 31 shared peripherals. Each of them has its own Peripheral Right Register (PRR) accessible within the SPBA memory-mapped registers, and consists of the Requesting Master Owner, the Resource Owner ID and the Resource Access Right fields.

SPBA memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
203_C000	Peripheral Rights Register (SPBA_PRR0)	32	R/W	0000_0007h	58.5.1/5017
203_C004	Peripheral Rights Register (SPBA_PRR1)	32	R/W	0000_0007h	58.5.1/5017
203_C008	Peripheral Rights Register (SPBA_PRR2)	32	R/W	0000_0007h	58.5.1/5017
203_C00C	Peripheral Rights Register (SPBA_PRR3)	32	R/W	0000_0007h	58.5.1/5017
203_C010	Peripheral Rights Register (SPBA_PRR4)	32	R/W	0000_0007h	58.5.1/5017
203_C014	Peripheral Rights Register (SPBA_PRR5)	32	R/W	0000_0007h	58.5.1/5017
203_C018	Peripheral Rights Register (SPBA_PRR6)	32	R/W	0000_0007h	58.5.1/5017
203_C01C	Peripheral Rights Register (SPBA_PRR7)	32	R/W	0000_0007h	58.5.1/5017
203_C020	Peripheral Rights Register (SPBA_PRR8)	32	R/W	0000_0007h	58.5.1/5017

Table continues on the next page...

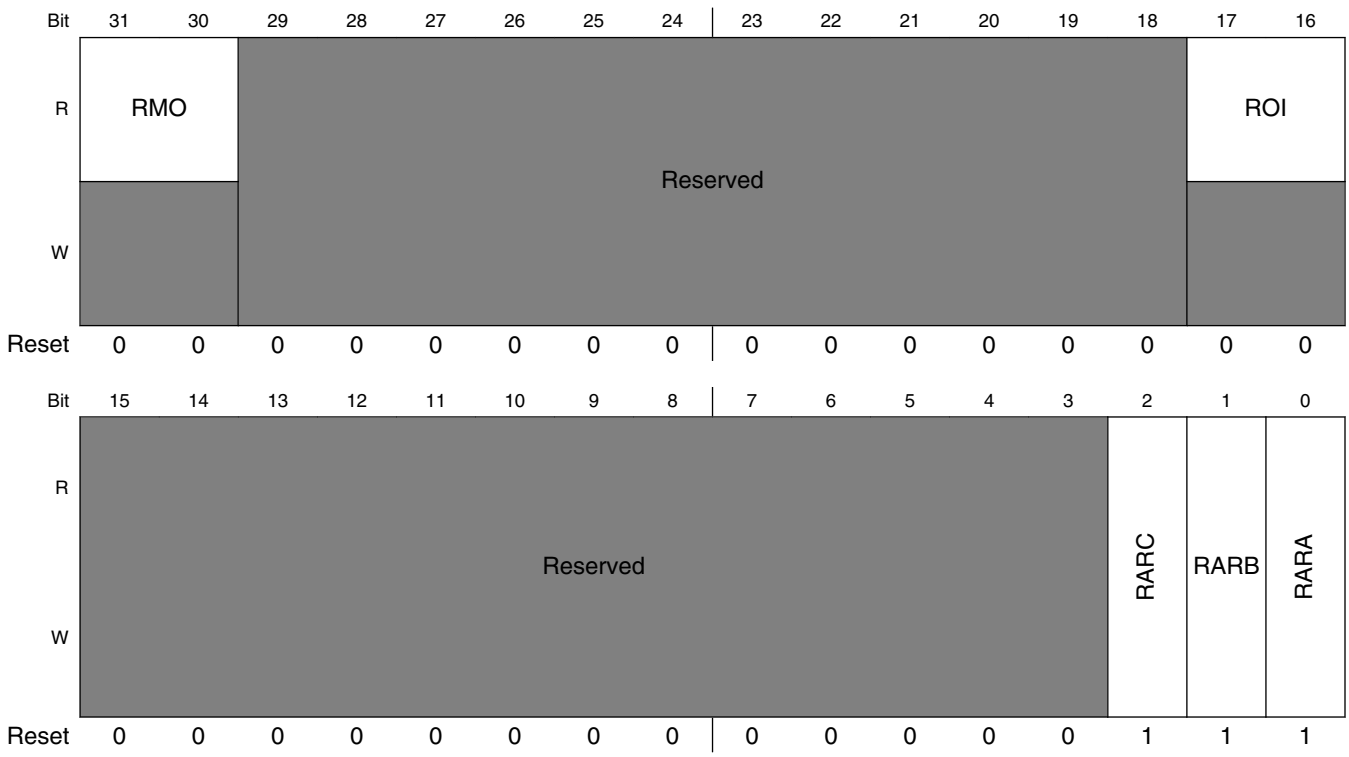
SPBA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
203_C024	Peripheral Rights Register (SPBA_PRR9)	32	R/W	0000_0007h	58.5.1/5017
203_C028	Peripheral Rights Register (SPBA_PRR10)	32	R/W	0000_0007h	58.5.1/5017
203_C02C	Peripheral Rights Register (SPBA_PRR11)	32	R/W	0000_0007h	58.5.1/5017
203_C030	Peripheral Rights Register (SPBA_PRR12)	32	R/W	0000_0007h	58.5.1/5017
203_C034	Peripheral Rights Register (SPBA_PRR13)	32	R/W	0000_0007h	58.5.1/5017
203_C038	Peripheral Rights Register (SPBA_PRR14)	32	R/W	0000_0007h	58.5.1/5017
203_C03C	Peripheral Rights Register (SPBA_PRR15)	32	R/W	0000_0007h	58.5.1/5017
203_C040	Peripheral Rights Register (SPBA_PRR16)	32	R/W	0000_0007h	58.5.1/5017
203_C044	Peripheral Rights Register (SPBA_PRR17)	32	R/W	0000_0007h	58.5.1/5017
203_C048	Peripheral Rights Register (SPBA_PRR18)	32	R/W	0000_0007h	58.5.1/5017
203_C04C	Peripheral Rights Register (SPBA_PRR19)	32	R/W	0000_0007h	58.5.1/5017
203_C050	Peripheral Rights Register (SPBA_PRR20)	32	R/W	0000_0007h	58.5.1/5017
203_C054	Peripheral Rights Register (SPBA_PRR21)	32	R/W	0000_0007h	58.5.1/5017
203_C058	Peripheral Rights Register (SPBA_PRR22)	32	R/W	0000_0007h	58.5.1/5017
203_C05C	Peripheral Rights Register (SPBA_PRR23)	32	R/W	0000_0007h	58.5.1/5017
203_C060	Peripheral Rights Register (SPBA_PRR24)	32	R/W	0000_0007h	58.5.1/5017
203_C064	Peripheral Rights Register (SPBA_PRR25)	32	R/W	0000_0007h	58.5.1/5017
203_C068	Peripheral Rights Register (SPBA_PRR26)	32	R/W	0000_0007h	58.5.1/5017
203_C06C	Peripheral Rights Register (SPBA_PRR27)	32	R/W	0000_0007h	58.5.1/5017
203_C070	Peripheral Rights Register (SPBA_PRR28)	32	R/W	0000_0007h	58.5.1/5017
203_C074	Peripheral Rights Register (SPBA_PRR29)	32	R/W	0000_0007h	58.5.1/5017
203_C078	Peripheral Rights Register (SPBA_PRR30)	32	R/W	0000_0007h	58.5.1/5017
203_C07C	Peripheral Rights Register (SPBA_PRR31)	32	R/W	0000_0007h	58.5.1/5017

58.5.1 Peripheral Rights Register (SPBA_PRR*n*)

This register controls master ownership and access for a peripheral.

Address: 203_C000h base + 0h offset + (4d × i), where i=0d to 31d



SPBA_PRR*n* field descriptions

Field	Description
31–30 RMO	Requesting Master Owner. This 2-bit register field indicates if the corresponding resource is owned by the requesting master or not. This register is reset to 2'b0 if ROI = 2'b0. 00 **UNOWNED** — The resource is unowned. 01 Reserved. 10 **ANOTHER_MASTER** — The resource is owned by another master. 11 **REQUESTING_MASTER** — The resource is owned by the requesting master.
29–18 -	This field is reserved. Reserved
17–16 ROI	Resource Owner ID. This field indicates which master (one at a time) can access to the PRR for rights modification. This is a read-only register. After reset, ROI bits are cleared ("00" -> un-owned resource).

Table continues on the next page...

SPBA_PRR*n* field descriptions (continued)

Field	Description
	A master performing a write access to the an un-owned PRR will get its ID automatically written into ROI, while modifying RARx bits. It can then read back the RMO, RAR, ROI bits to make sure RMO returns the right value, ROI bits contain its ID and RARx bits are correctly asserted. Then no other master (whom ID is different from the one stored in ROI) will be able to modify RAR fields. Owner master of a peripheral can assert its dead_owner signal, or write 1'b0 in the RARx to release the ownership (ROI[1:0] reset to 2'b0). 00 **UNOWNED** — Unowned resource. 01 **MASTER_A** — The resource is owned by master A port. 10 **MASTER_B** — The resource is owned by master B port. 11 **MASTER_C** — The resource is owned by master C port.
15–3 -	This field is reserved. Reserved
2 RARC	Resource Access Right. Control and Status bit for master C. This field indicates whether master C can access the peripheral. From 0 up to 3 masters can have permission to access a resource (all the master can be granted on a peripheral, but only one access at a time will be granted by SPBA). 0 **PROHIBITED** — Access to peripheral is not allowed. 1 **ALLOWED** — Access to peripheral is granted.
1 RARB	Resource Access Right. Control and Status bit for master B. This field indicates whether master B can access the peripheral. From 0 up to 3 masters can have permission to access a resource (all the master can be granted on a peripheral, but only one access at a time will be granted by SPBA). 0 **PROHIBITED** — Access to peripheral is not allowed. 1 **ALLOWED** — Access to peripheral is granted.
0 RARA	Resource Access Right. Control and Status bit for master A. This field indicates whether master A can access the peripheral. From 0 up to 3 masters can have permission to access a resource (all the master can be granted on a peripheral, but only one access at a time will be granted by SPBA). 0 **PROHIBITED** — Access to peripheral is not allowed. 1 **ALLOWED** — Access to peripheral is granted.

Chapter 59
Sony/Philips Digital Interface (SPDIF)

59.1 Introduction

The Sony/Philips Digital Interface (SPDIF) audio block is a stereo transceiver that allows the processor to receive and transmit digital audio.

The SPDIF transceiver allows the handling of both SPDIF channel status (CS) and User (U) data and includes a frequency measurement block that allows the precise measurement of an incoming sampling frequency.

A recovered clock is provided to drive both internal and external components in the system such as ESAI ports, as well as external A/Ds or D/As, with clocking control provided via related registers.

As the SPDIF internal data width is 24-bit, the eight most-significant bits of all registers return zeros.

The figure below shows a block diagram of the SPDIF transceiver data paths (receiver and transmitter) and its interface.

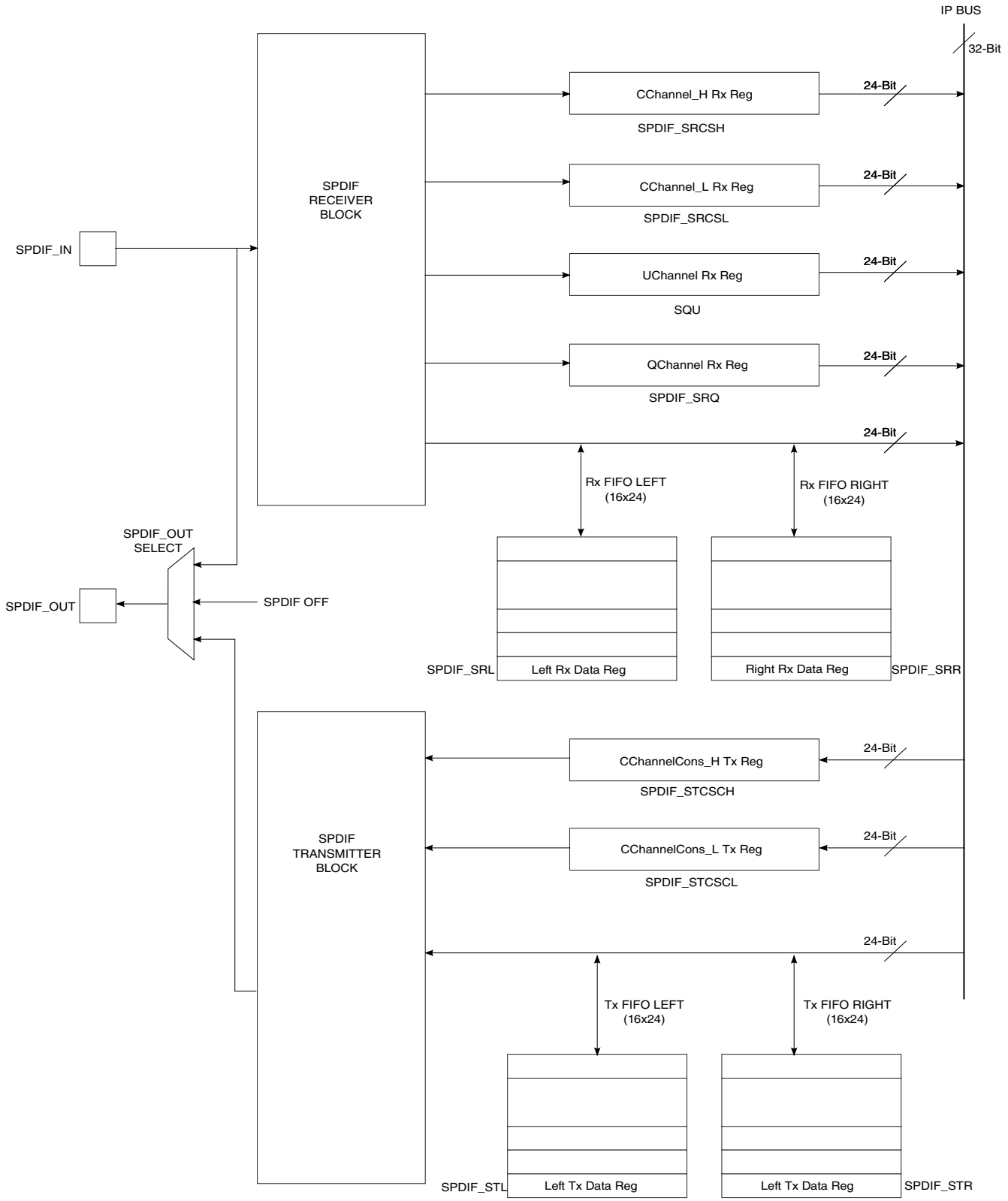


Figure 59-1. SPDIF Transceiver Data Interface Block Diagram

59.1.1 Overview

The SPDIF is composed of two parts: SPDIF Receiver and SPDIF Transmitter.

The SPDIF receiver extracts the audio data from each SPDIF frame and places the data in the SPDIF Rx left and right FIFOs. The Channel Status and User Bits are also extracted from each frame and placed in the corresponding registers. The SPDIF receiver also provides a bypass option for direct transfer of the SPDIF input signal to the SPDIF transmitter.

For the SPDIF transmitter, the audio data is provided by the processor via the SPDIFTxLeft and SPDIFTxRight registers. The Channel Status bits are also provided via the corresponding registers. The SPDIF transmitter generates a SPDIF output bitstream in the biphase mark format (IEC60958), which consists of audio data, channel status and user bits.

In the SPDIF transmitter, the IEC60958 biphase bit stream is generated on both edges of the SPDIF Transmit clock. The SPDIF Transmit clock is generated by the SPDIF internal clock generate block and the sources are from outside of the SPDIF block. For the SPDIF receiver, it can recover the SPDIF Rx clock. Both the Rx clock and Tx clock are sent to the ASRC.Figure 59-2 shows the clock structure of the SPDIF transceiver.

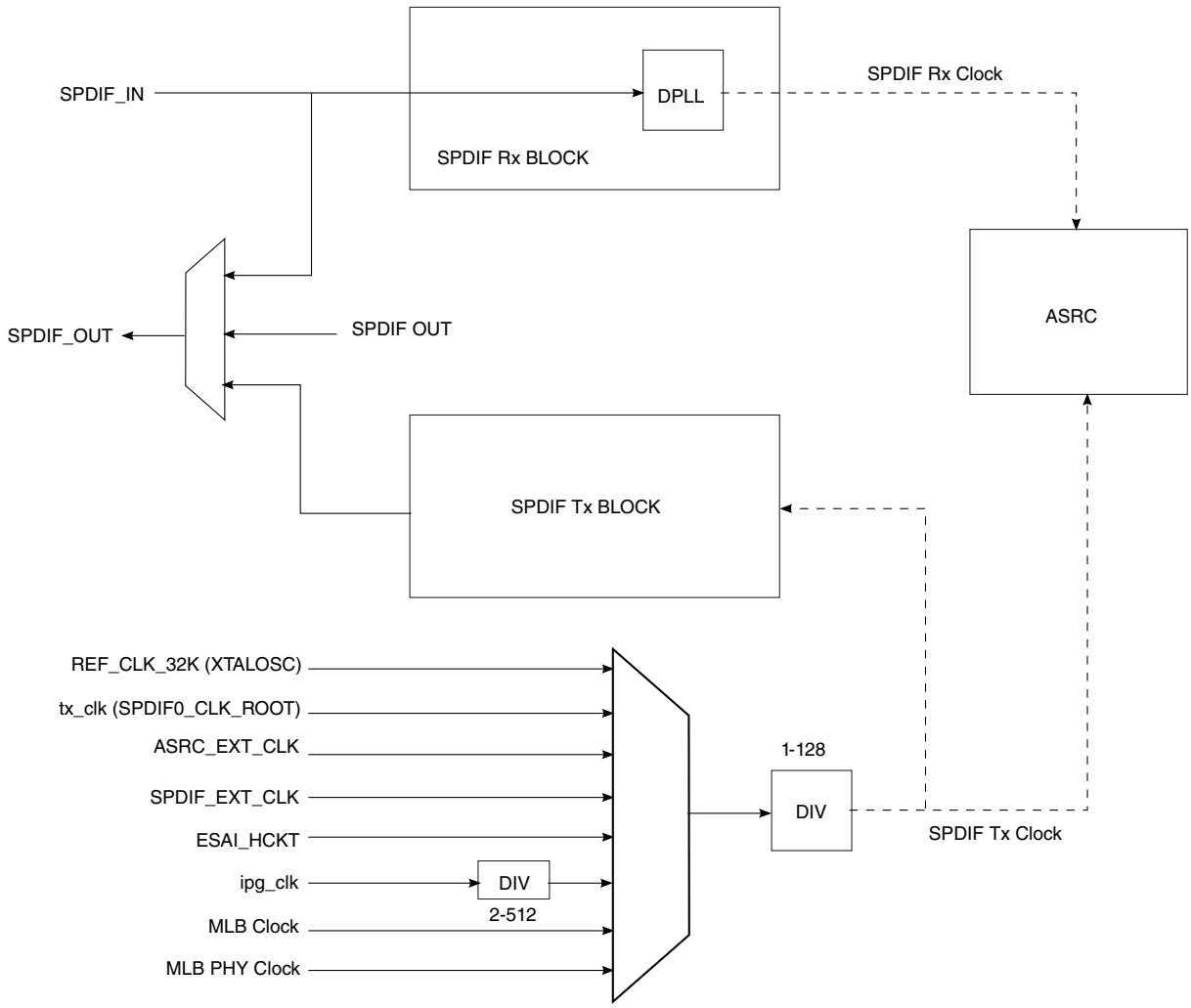


Figure 59-2. SPDIF Transceiver Clock Diagram

59.2 External Signals

Table 59-1. SPDIF External Signals

Signal	Description	Pad	Mode	Direction
SPDIF_EXT_CLK	External clock signal	ENET_CRS_DV	ALT3	I
		RGMII_TXC	ALT2	
SPDIF_IN	Input line	EIM_D21	ALT7	I
		ENET_RX_ER	ALT3	
		GPIO_16	ALT4	
		KEY_COL3	ALT6	

Table continues on the next page...

Table 59-1. SPDIF External Signals (continued)

Signal	Description	Pad	Mode	Direction
SPDIF_LOCK	Lock signal	ENET_MDIO	ALT6	O
		GPIO_7	ALT6	
SPDIF_OUT	Output line signal	EIM_D22	ALT6	O
		ENET_RXD0	ALT3	
		GPIO_17	ALT4	
		GPIO_19	ALT2	
SPDIF_SR_CLK	SR Lock signal	ENET_REF_CLK	ALT6	O
		GPIO_8	ALT6	

59.3 Clocks

The table found here describes the clock sources for SPDIF.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 59-2. SPDIF Clocks

Clock name	Clock Root	Description
gclkw_t0	ipg_clk_root	Global clock
ipg_clk_s	ipg_clk_root	Peripheral access clock
tx_clk	spdif0_clk_root	Module Tx clock

59.4 Functional Description

59.4.1 SPDIF Receiver

The SPDIF receiver extracts the audio data from each SPDIF frame and places the data in Rx left and right FIFOs.

The Tx left and right FIFOs are 16-deep and 24-bit-wide (equal to the audio data width). The Channel Status and User Bits are also extracted from each frame and placed in corresponding registers. The SPDIF receiver also provides a bypass option for direct transfer of the SPDIF input signal to the SPDIF transmitter.

The SPDIF receiver handles the main data audio stream and recovers the bit clock from the SPDIF input signal. The sample rate can be determined from the frequency measuring block. Additionally, the receiver supports the SPDIF C and U channels. The SPDIF C and U channel data is interfaced directly to memory-mapped registers.

All the data registers are controlled by the Interrupt Control Block and transferred to the memory-mapped IP bus.

The following functions are performed by the SPDIF receiver:

- Audio Data Reception see Audio Data Reception
- Channel Status bits Reception see Channel Status Reception
- U Channel bits Reception see User Bit Reception
- Validity Flag Reception see Validity Flag Reception
- SPDIF Receiver Exception support see SPDIF Receiver
- SPDIF Lock Detection

59.4.1.1 Audio Data Reception

The SPDIF Receiver block extracts the audio data from the IEC60958 stream, and outputs this via Rx left and right FIFOs to the memory-mapped registers SPDIFRxLeft and SPDIFRxRight.

Data from the SPDIF receiver is buffered in receive FIFO, and can be read by the processor from the memory-mapped registers.

- **SPDIF receiver data registers - Behavior on overrun, underrun**

The SPDIF Data Receive registers (SPDIFRxLeft and SPDIFRxRight) have individual FIFOs for left and right channel. As a result, there is always the possibility that left and right FIFOs may go out of sync due to FIFO underruns and FIFO overruns that affect only one part (left or right) of any FIFOs. To prevent this from happening, hardware has been added to the device. Two mechanisms to prevent mismatch between the FIFOs are available.

If a SPDIF Data Rx FIFO overrun occurs on e.g. the right half of the FIFO, the sample that caused the overrun is not written to the right half (due to overrun). Special hardware will make sure the next sample is not written to the left half of the FIFO. If the overrun occurs on the left half of the FIFO, the next sample is not written to the right half of the FIFO.

- **SPDIF receiver data registers - Automatic resynchronization of FIFOs**

An automatic FIFO resynchronization feature is available. It can be enabled and disabled separately for every FIFO. If it is enabled, the hardware will check to see if the left and right FIFOs are in sync. If that is not the case, it will set the filling pointer of the right FIFO to be equal to the filling pointer of the left FIFO.

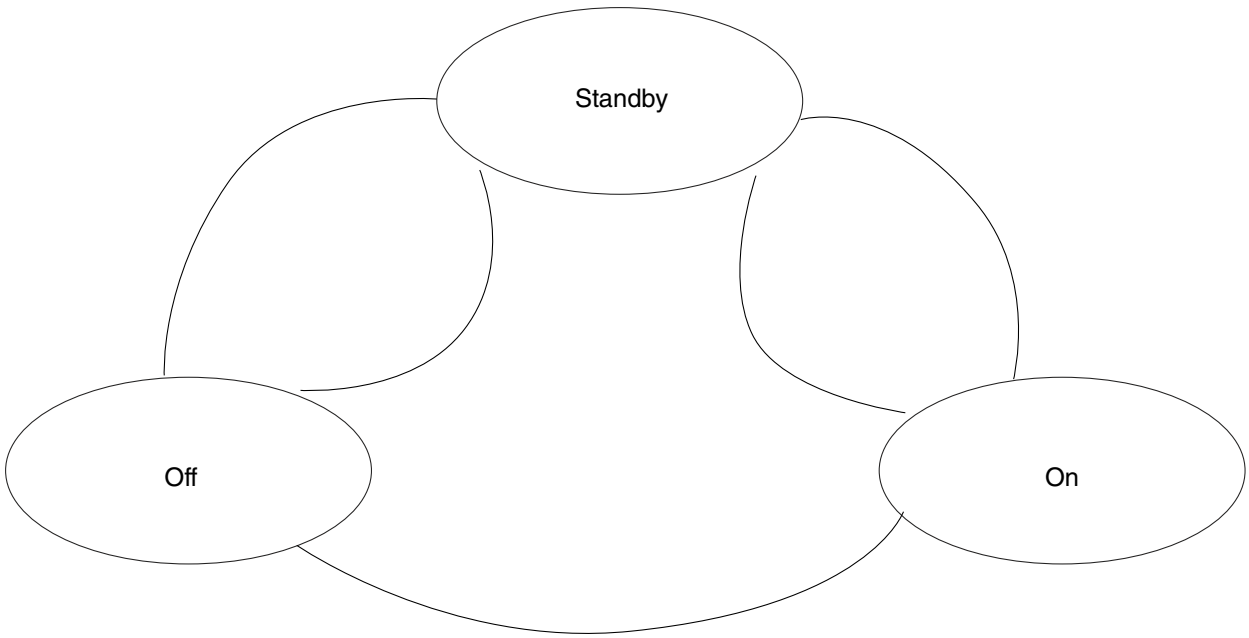


Figure 59-3. FIFO Auto-resync Controller State Machine

The operation is explained from the state diagram shown above. Every FIFO auto-resync controller has a state machine with 3 states: Off, StandBy and On. In the On state, the filling of the left FIFO is compared with the filling of right, and if they are not equal, right is made equal to left, and an interrupt is generated.

The controller will stay in Off state when the feature is disabled. When not disabled, the state machine will go to Off state on any processor read or write to the FIFO. It will go from On or Off to Standby on any left sample read from SPDIF Tx FIFOs, or on any left sample write to SPDIF Rx FIFOs. The controller will go from Standby to On on any right sample read from SPDIF Tx FIFO, or on any right sample write to SPDIF Rx FIFO. There is a control bit in the SPDIFConfig register to enable/disable the feature for the SPDIF Rx FIFO and SPDIF Tx FIFO.

59.4.1.1.1 Application Note

The automatic FIFO resynchronization can be switched on, and will avoid all mismatches between left and right FIFOs, if the software obeys the following rules: 1. When the left data is read or written to the left FIFO, in the same place of the program, data must be read or written to the right FIFO. Maximum time difference between left and right is 1/2 sample clock. (E.g. if sample frequency is 44 Khz, approximately 10 micro-seconds. For 88 Khz, approximately 5 micro-seconds.) 2. Write/read data to FIFO s at least 2 samples at the time. If there is a mismatch Left-Right, the resync logic may go on only 1 sample clock after last data is read/written to the FIFO. Also acceptable is polling the FIFO, if at least part of the time 2 samples will be read/written to it.

- **SPDIF receiver - Additional features**

There are three exceptions associated with the SPDIF Receivers FIFOs

- full
- under/overrun
- resync

When the "full" condition is set for processor data input registers, the processor should read data from the FIFO, before overrun occurs. When "full" is set, and the FIFO contains e.g. 6 samples, it is acceptable for the software to read first 6 samples from the LEFT address, followed by 6 samples from the RIGHT address, or 6 samples from the RIGHT address, followed by 6 samples from the LEFT address, or 1 sample LEFT, followed by 1 sample RIGHT repeated 6 times. There is no order specified.

The implementation for SPDIF Rx is a double FIFO, one for left and one for right. "full" is set when both FIFOs are full. "underrun, overrun" are set when one of the FIFOs do underrun or do overrun. The resync interrupt means hardware took special action to resynchronize left and right FIFOs.

The FIFO level at which the "full" interrupt is generated, is programmable via the Full Select field in the SPDIFConfigReg register.

Rx FIFO on and Rx FIFO reset.

Two additional control fields of the SPDIF Rx FIFO are the on/off select and FIFO reset fields.

If on/off select is set to off, all-zero will be read from the FIFO, irrespective of the data received over the SPDIF interface.

If FIFO reset is set, the FIFO is blocked at "1 sample in FIFO". In this, the full interrupt will be on if FullSelect is set to "00". If FullSelect is set to any other value, interrupt will be off. The other interrupts are always off.

59.4.1.2 Channel Status Reception

A total of 48 channel status bits are received in two registers. No interpretation is performed by the SPDIF receiver block.

Channel Status Bits are ordered first bit left. CS-channel MSB bit "0" is located in bit position 23 in the memory-mapped register SPDIFRxCChannel_h. CS-channel bit "23" is considered the LSB bit 0 in the register. C-channel bit 24 to 47 is seen as [23:0] bits of register SPDIFRxCChannel_l.

59.4.1.2.1 Channel Status Interrupt

When the value of a new SPDIF "CS" channel status frame is loaded in the register, an interrupt is generated. The interrupt is cleared when the processor writes the corresponding bit in the InterruptStat register.

59.4.1.3 User Bit Reception

There are two modes for U Channel reception, CD and non-CD. As is decided by USyncMode (bit 1 of CDText_Control register).

- **Behavior of U Channel receive interface on incoming CD U Channel Sub-code in SPDIF receiver**.

This mode is selected if UsyncMode, bit 1 in register CD Text control is set "1".

The CD sub-code stream embedded into the SPDIF U channel consists of a sequence of packets. Every packet is made up 98 "symbols". The first two symbols of every packet are "sync symbols", the other 96 symbols are "data symbols".

Any sequence found in the SPDIF U channel stream starting with a leading one, followed by 7 information bits, is recognized as a "data symbol". Subsequent data symbols are separated by "pauses". During the "pause", "zero bits" are seen on the SPDIF U channel.

Data symbols are coming in MSB first. The MSB is the leading one.

When a "long pause" is seen between 2 subsequent "data symbols", the SPDIF receiver will assume the reception of one or more "sync symbols". Table below gives details.

Table 59-3. Sync Control Bits

Number of U Channel zero bits	Corresponding number of sync symbols
0-1	Unpredictable, not allowed

Table continues on the next page...

Table 59-3. Sync Control Bits (continued)

Number of U Channel zero bits	Corresponding number of sync symbols
2-10	0
11-22	1
23-34	2
35-46	3
>45	Unpredictable, not allowed

The recognition of the number of sync symbols derives from the fact that the U channel transmitter in the CD channel decoder will transmit one symbol on average every 12 SPDIF channel bits. On this average rate, there is a maximum tolerance of 5%.

The SPDIF receiver is tolerant of symbol errors. Due to the physical nature of the transmission of the data over the CD disc, not more than 1 out of any 5 consecutive user channel symbols may be in error. The error may cause a change in data value, which is not detected by this interface, or it may cause a data symbol to be seen as a sync symbol, or a sync symbol to be seen as a data symbol. However, not more than 1 out of any 5 consecutive user channel symbols should be affected in this way.

The SPDIF U channel circuitry recognizes the 98-symbol packet structure, and sends the 96 symbol payload to the processor application. The 96 symbol payload is transmitted to the processor via 2 registers:

- The SPDIFRxUChannel register. In this register, data is presented 3 symbols at the time to the processor. Every time 3 new valid symbols, received on the SPDIF U Channel are present, the UChannelRxFull interrupt is asserted. For one 98-symbol packet, 96 symbols are carried across SPDIFRxUChannel. To transfer all this data, 32 UChannelRxFull interrupts are generated.
- The QChannelReceive register. In this register, only the Q bit of the packet is accumulated.Operation is similar to UChannelReceive. Because only Q-bit is transferred, only 96 Q-bits are transferred for any 98-symbol packet. To transfer this data, 4 QChannelRxFull interrupts are generated. When QChannelRxFull occurs, it is coincident with UChannelRxFull. There is only one QChannelRxFull for every 8 UChannelRxFull. The convention is that most significant data is transmitted first, and is left-aligned in the registers.
- Timing regarding packet boundary is extracted by hardware. The last UChannelRxFull corresponding to a given packet should be coincident with the last QChannelRxFull. In this last U, Q channel interrupt, symbols 95-98 are received, Q channel bits 67-98. The interrupts are coincident with UQSyncFound, flagging last symbols of the current frame.

- When the start of the new packet is found before the current packet is complete (less than 98 symbols in the packet), the UQFrameError interrupt is set. The application software should read out UChannelReceive and QchannelReceive registers, discard the value, and assume the start of a new packet.
- As already said, packet sync extraction is tolerant for single-symbol errors. Packet sync detection is based on the recognition of the sequence data-sync-sync-data in the symbol stream, because this is the only syncing sequence that is not affected by single errors. If the sync symbols are not found 98 symbols after the previous occurrence, it is assumed to be destroyed by channel error, and a new sync symbols is interpolated.
- Normally, only data bytes are passed to the application software. Every databyte will have its most significant bit set. If sync symbols are passed to the application software, they are seen as all-zero symbols. Sync symbols can only end up in the data stream due to channel error.

- **Behavior of U Channel receive interface on incoming non-CD data**.

This mode is selected if UsyncMode, bit 1 in register CD Text control is set'0'.

In non-CD mode, the SPDIF U channel stream is recognized as a sequence of "data symbols". No packet recognition is done.

Any sequence found in the SPDIF U channel stream starting with a leading one, followed by 7 information bits, is recognized as a "data symbol". Subsequent data symbols are separated by "pauses". During the "pause", "zero bits" are seen on the SPDIF U channel.

3 consecutive data symbols seen in the SPDIF U Channel stream are grouped together into the SPDIFRxUChannel register. First symbol is left, last symbol is right aligned. When SPDIFRxUChannel contains 3 new data symbols, UChannelRxFull is asserted.

In this mode, the operation of QchannelRx and associated interrupt QchannelRxFull is reserved, undefined. And the operation of UQFrameError and UQSyncFound is also reserved, undefined.

The U channel is extracted, and output by the SPDIF Rx on SPDIFRxUChannel-Stream.

When incoming SPDIF data parity error or bit error is detected, and if the next SPDIF word for that channel is error-free, the SPDIF word in error is replaced with the average of the previous word and next word. When incoming SPDIF data parity error or bit error is detected, and the next SPDIF word is in error, the previous SPDIF word is repeated.

59.4.1.4 Validity Flag Reception

An interrupt is associated with the Validity flag. (interrupt 16 - SPDIFValNoGood). This interrupt is set every time a frame is seen on the SPDIF interface with the validity bit set to "invalid".

59.4.1.5 SPDIF Receiver Interrupt Exception Definition

Several SPDIF exceptions can trigger an interrupt.

They are:

- Control Status channel change. Set when SPDIFRxCChannel_l register is updated. The register is updated for every new C-Channel received. The exception is reset on write to InterruptClear register.
- SPDIF Illegal Symbol. Set on reception of illegal symbol during SPDIF receive. Reset by writing register InterruptClear.[1]
- SPDIF bit error. Set on reception of bit error. (Parity bit does not match). Reset on write to InterruptClear register.
- Receive data FIFO full. Set when SPDIF receive data FIFO is full.
- Receive data FIFO underrun/overrun. Set when there is a underrun/overrun on the SPDIF receive data FIFO.
- Receive data FIFO resynchronization. Set when a resynchronization event occurs on the SPDIF receive data FIFO.
- Receive U Channel buffer full. Set when next 24 bits of U channel code are available.
- Receive Q Channel buffer overrun. Set when Q channel buffer overrun.
- Receive U Channel buffer overrun. Set on U channel buffer overrun.
- Receive Q Channel buffer full. Set when next 24 bits of Q channel code are available.
- Receive UQ sync found. Set when UQ channel sync found.
- Receive UQ frame error. Set when UQ frame error found.

1. The SPDIF input is a biphase/mark modulated signal. The time between any two successive transitions of the SPDIF signal is always 1, 2 or 3 SPDIF symbol periods long. The SPDIF receiver will parse the stream, and split it in so-called symbols. It recognizes s1, s2 and s3 symbols, depending on the length of the symbols. Not all sequences of these symbols are allowed. To give an example, a sequence s2-s1-s1-s1-s2 cannot occur in a no-error SPDIF signal. If the receiver finds such an illegal sequence, the illegal symbol interrupt is set. No corrective action is undertaken. When the interrupt occurs, this means that(a) The SPDIF signal is destroyed by noise (b) The SPDIF frequency changed.

59.4.1.6 Standards Compliance

The SPDIF interface is compatible with the Tech 3250-E standard of the European Broadcasting Union, except clause 6.3.3 and the IEC60958-3 Ed2 for relevant topics.

Supported input frequency range is 12 Khz up to 96 Khz. (fully compliant) and 96 Khz up to 176 Khz (Can interface with compliant SPDIF transmitter within same cabinet, making reasonable assumptions on jitter added due to interconnecting wire.)

Tolerated jitter on SPDIF input signals are 0.25 bit peak-peak for high frequencies. There is no jitter limit for low frequencies. The user channel extraction in CD mode is capable of coping with single-symbol errors, and still retrieve U channel frames on correct boundaries. This capability is required for reliable reception of CD-Text from some Philips CD channel decoders. This capability was deemed more important than compliance with the IEC60958 annex A.3 standard, and for this reason user channel reception is not compliant with IEC60958 annex A.3. However, the interface is capable to receive U channel inserted by a typical CD channel decoder. Also, in this case, it is more robust and tolerant for channel error than what is required by IEC60958 annex A.3.

59.4.1.7 SPDIF PLOCK Detection and Rxclk Output

Using the high speed system clock, the internal DPLL can extract the bit clock (advanced pulse) from the input bitstream. When this internal DPLL is locked, the LOCK bit of PhaseConfig Register will be set, and the SPDIF Lock output pin SPDIF_LOCK will be asserted.

After DPLL has locked, the pulses are generated, and the average pulse rate is 128 x the sampling frequency.(For a 44.1 Khz input sampling frequency, the average pulse rate = 128 x 44.1 Khz.) The pulse signal is used in the FreqMeas circuit to generate the frequency measurement result.

59.4.1.8 Measuring Frequency of SPDIF_RxClk

The internal DPLL can extract the bit clock (advanced plus) from the input bitstream. To do that, it is necessary to measure the frequency of the incoming signal in relationship with the system clock (BUS_CLK).

Associated with it are two registers, PhaseConfig and FreqMeas. The circuit will measure the frequency of the incoming clock as a function of the BUS_CLK. The circuit is a second-order filter. The output is a value represented by an unsigned number stored in the 24-bit FreqMeas register, giving the frequency of the source as a function of the BUS_CLK.

FreqMeas[23:0] = FreqMeas_CLK / BUS_CLK * 2^{10} * GAIN.

For example, if the GAIN is selected as 8*(2**10) (PhaseConfig[5:3] = 3'b011), the actual result

FreqMeas_CLK / BUS_CLK is equal to FreqMeas[23:0] / 2^{23}.

59.4.2 SPDIF Transmitter

Audio data for the SPDIF transmitter is provided by processor via the SPDIFTxLeft and SPDIFTxRight registers.

Clocking for SPDIF transmitter is selected through a multiplexer from several clock sources (see TxClk_Source for clock source inputs). The SPDIF transmitter clock source can be divided down as needed using Txclk_DF. The SPDIF transmitter output can be chosen from either the SPDIF transmitter block, directly from the SPDIF receiver (via the output multiplexer), or disabled.

The SPDIF transmitter generates a SPDIF output bitstream in IEC60958 biphase mark format, consisting of audio data, channel status.

59.4.2.1 Audio Data Transmission

Audio data for the SPDIF transmitter is provided by the processor via SPDIFTxLeft and SPDIFTxRight registers. They send audio data to Tx left and right FIFOs. The Tx left and right FIFOs are also 16-deep and 24-width (equal to the audio data width).

- **SPDIF transmitter data registers - Behavior on overrun, underrun**

The SPDIF Data Transmit registers (SPDIFTxLeft and SPDIFTxRight) have individual FIFOs for left and right channel. As a result, there is always the possibility that left and right FIFOs may go out of sync due to FIFO underruns and FIFO overruns that affect only one part (left or right) of any FIFO. To prevent this from happening, hardware has been added on the device. Two mechanisms to prevent mismatch between the FIFOs are available.

If SPDIF Tx FIFO underruns on the right half of the FIFO, no sample leaves that FIFO (because it was already empty). Special hardware will make sure that the next sample read from the left FIFO will not leave the FIFO (no read strobe is generated). If the underrun occurs on the left half of the FIFO, next read strobe to the right FIFO is blocked.

- **SPDIF transmitter data registers - Automatic resynchronization of FIFOs**

See Audio Data Reception.

- **SPDIFTxLeft, SPDIFTxRight details**

With SPDIF Tx FIFOs three exceptions are associated.

- empty
- under/overrun
- resync

When the empty condition is set for processor data output registers, the processor should write data to the FIFO, before underrun occurs. When empty is set and, for instance, 6 samples need to be written, it is acceptable for the software to write first 6 samples from the LEFT address, followed by 6 samples from the RIGHT address, or 1 sample LEFT, followed by 1 sample RIGHT repeated 6 times. Left should be written before right. The implementation of all data out FIFOs is a double FIFO, one for left and one for right. Empty is set when both FIFOs are empty. Underrun, overrun are set when one of the FIFOs do underrun or do overrun. Resync is set when the hardware resynchronizes left and right FIFOs.

On receiving underrun, overrun interrupt, synchronization between Left and Right words in the FIFOs may be lost. Synchronization will not be lost when the underrun or overrun comes from the IEC60958 side of the FIFO. If the processor reads or writes more data from, for example, left than from right, synchronization will be lost. If automatic resynchronization is enabled, and if the software obeys the rules to let this work, resynchronization will be automatic.

59.4.2.2 Channel Status Transmission

A total of 48 Consumer channel status bits are transmitted from two registers. Channel Status Bits are ordered first bit left.

CS-channel MSB bit "0" is located in bit position 23 in the memory-mapped register SPDIFTxCChannelCons_h. CS-channel bit "23" is considered bit 0 in the register. C-channel bits 24-47 are seen as MSB-LSB bits of register SPDIFTxCChannelCons_l.

59.4.2.3 Validity Flag Transmission

The validity bit setting is performed via bit 5 of the SPDIF_SCR register.

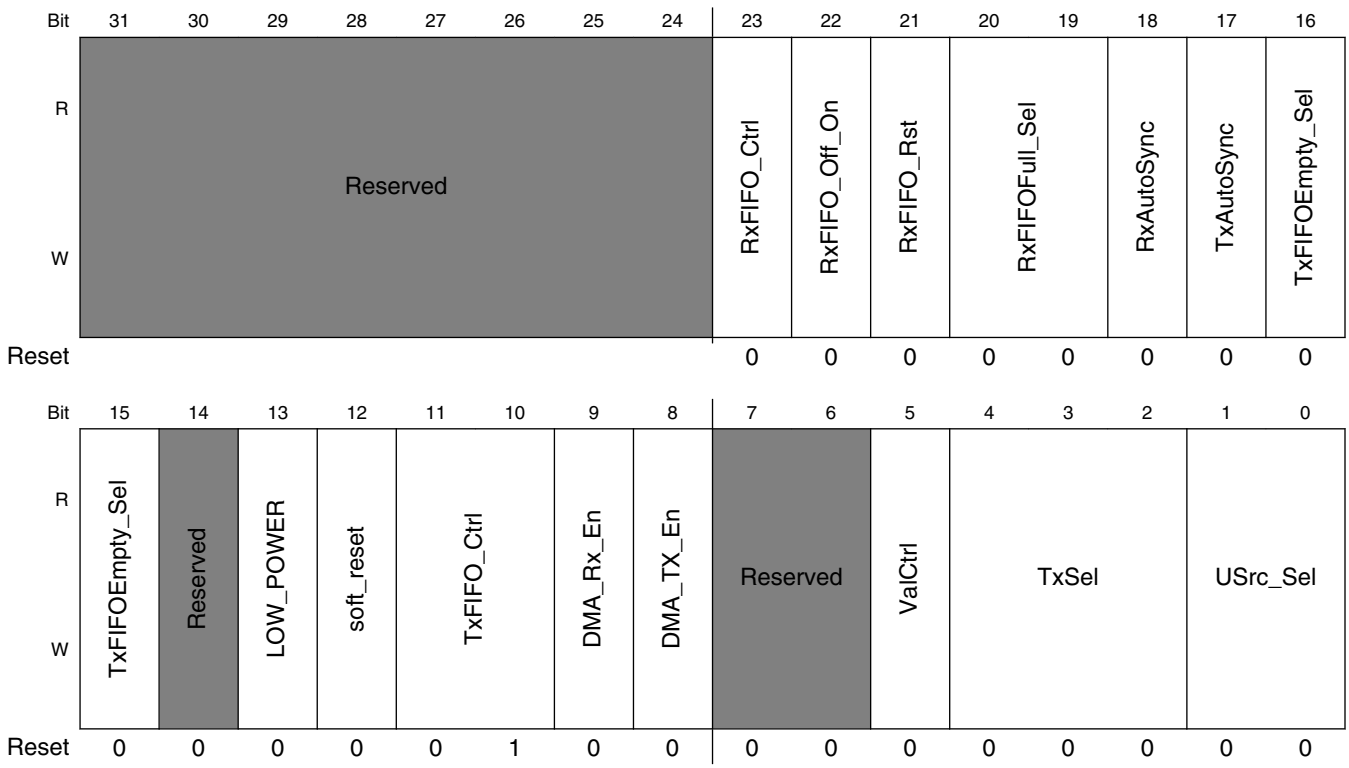
59.5 SPDIF Memory Map/Register Definition

SPDIF memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
200_4000	SPDIF Configuration Register (SPDIF_SCR)	32	R/W	0000_0400h	59.5.1/5035
200_4004	CDText Control Register (SPDIF_SRCD)	32	R/W	0000_0000h	59.5.2/5037
200_4008	PhaseConfig Register (SPDIF_SRPC)	32	R/W	0000_0000h	59.5.3/5038
200_400C	InterruptEn Register (SPDIF_SIE)	32	R/W	0000_0000h	59.5.4/5039
200_4010	InterruptStat Register (SPDIF_SIS)	32	R	0000_0002h	59.5.5/5041
200_4010	InterruptClear Register (SPDIF_SIC)	32	W	0000_0000h	59.5.6/5043
200_4014	SPDIFRxLeft Register (SPDIF_SRL)	32	R	0000_0000h	59.5.7/5044
200_4018	SPDIFRxRight Register (SPDIF_SRR)	32	R	0000_0000h	59.5.8/5045
200_401C	SPDIFRxCChannel_h Register (SPDIF_SRCSH)	32	R	0000_0000h	59.5.9/5045
200_4020	SPDIFRxCChannel_l Register (SPDIF_SRCSL)	32	R	0000_0000h	59.5.10/ 5046
200_4024	UchannelRx Register (SPDIF_SRU)	32	R	0000_0000h	59.5.11/ 5046
200_4028	QchannelRx Register (SPDIF_SRQ)	32	R	0000_0000h	59.5.12/ 5047
200_402C	SPDIFTxLeft Register (SPDIF_STL)	32	W	0000_0000h	59.5.13/ 5047
200_4030	SPDIFTxRight Register (SPDIF_STR)	32	W	0000_0000h	59.5.14/ 5048
200_4034	SPDIFTxCChannelCons_h Register (SPDIF_STCSCH)	32	R/W	0000_0000h	59.5.15/ 5048
200_4038	SPDIFTxCChannelCons_l Register (SPDIF_STCSCL)	32	R/W	0000_0000h	59.5.16/ 5049
200_4044	FreqMeas Register (SPDIF_SRFM)	32	R	0000_0000h	59.5.17/ 5049
200_4050	SPDIFTxClk Register (SPDIF_STC)	32	R/W	0002_0F00h	59.5.18/ 5050

59.5.1 SPDIF Configuration Register (SPDIF_SCR)

Address: 200_4000h base + 0h offset = 200_4000h



SPDIF_SCR field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23 RxFIFO_Ctrl	0 Normal operation 1 Always read zero from Rx data register
22 RxFIFO_Off_On	0 SPDIF Rx FIFO is on 1 SPDIF Rx FIFO is off. Does not accept data from interface
21 RxFIFO_Rst	0 Normal operation 1 Reset register to 1 sample remaining
20–19 RxFIFOFull_Sel	00 Full interrupt if at least 1 sample in Rx left and right FIFOs 01 Full interrupt if at least 4 sample in Rx left and right FIFOs 10 Full interrupt if at least 8 sample in Rx left and right FIFOs 11 Full interrupt if at least 16 sample in Rx left and right FIFO
18 RxAutoSync	0 Rx FIFO auto sync off 1 RxFIFO auto sync on
17 TxAutoSync	0 Tx FIFO auto sync off 1 Tx FIFO auto sync on

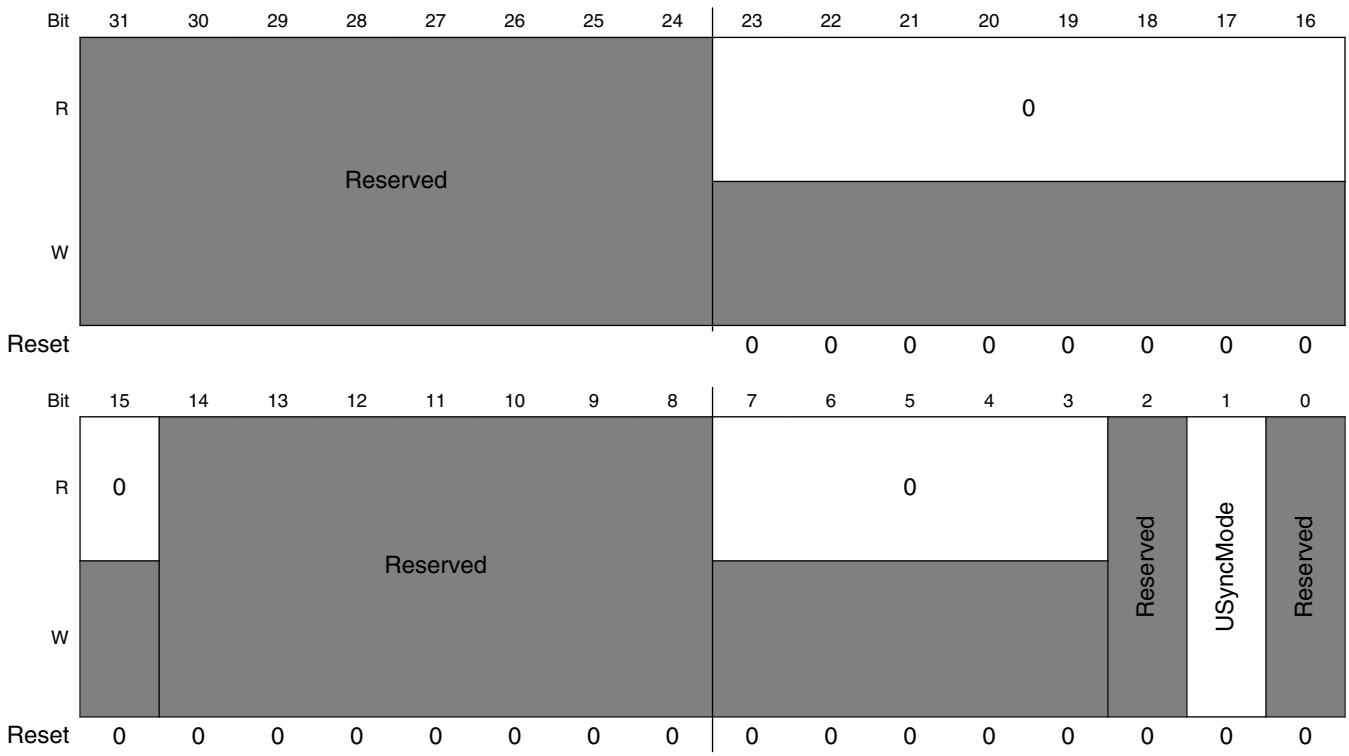
Table continues on the next page...

SPDIF_SCR field descriptions (continued)

Field	Description
16–15 TxFIFOEmpty_Sel	00 Empty interrupt if 0 sample in Tx left and right FIFOs 01 Empty interrupt if at most 4 sample in Tx left and right FIFOs 10 Empty interrupt if at most 8 sample in Tx left and right FIFOs 11 Empty interrupt if at most 12 sample in Tx left and right FIFOs
14 -	This field is reserved. Reserved
13 LOW_POWER	When write 1 to this bit, it will cause SPDIF enter low-power mode. return 1 when SPDIF in Low-Power mode.
12 soft_reset	When write 1 to this bit, it will cause SPDIF software reset. The software reset will last 8 cycles. When in the reset process, return 1 when read. else return 0 when read.
11–10 TxFIFO_Ctrl	00 Send out digital zero on SPDIF Tx 01 Tx Normal operation 10 Reset to 1 sample remaining 11 Reserved
9 DMA_Rx_En	DMA Receive Request Enable (RX FIFO full)
8 DMA_TX_En	DMA Transmit Request Enable (Tx FIFO empty)
7–6 -	This field is reserved. Reserved
5 ValCtrl	0 Outgoing Validity always set 1 Outgoing Validity always clear
4–2 TxSel	000 Off and output 0 001 Feed-through SPDIFIN 101 Tx Normal operation Others Reserved
USrc_Sel	00 No embedded U channel 01 U channel from SPDIF receive block (CD mode) 10 Reserved 11 U channel from on chip transmitter

59.5.2 CDText Control Register (SPDIF_SRCD)

Address: 200_4000h base + 4h offset = 200_4004h

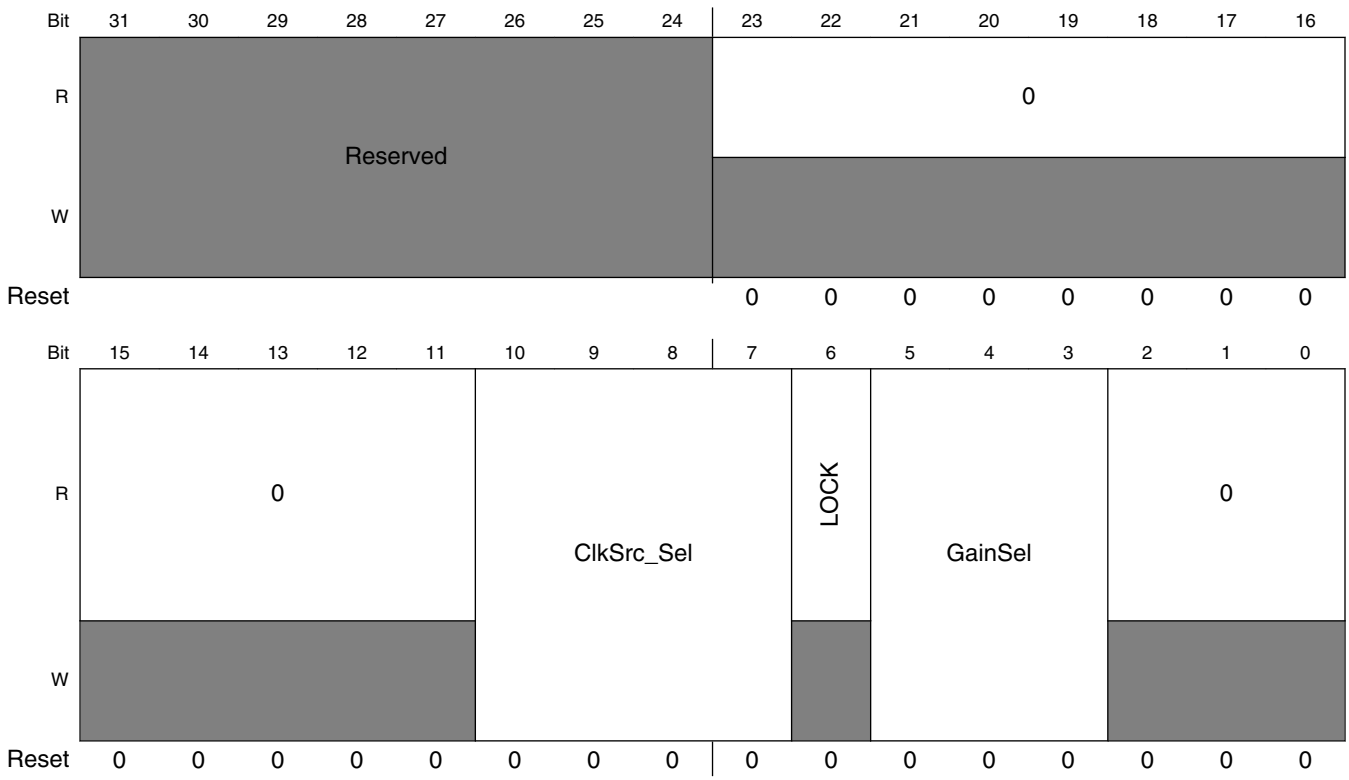


SPDIF_SRCD field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23–15 Reserved	This read-only field is reserved and always has the value 0.
14–8 -	This field is reserved. Reserved. set to zero.
7–3 Reserved	This read-only field is reserved and always has the value 0.
2 -	This field is reserved. Reserved.
1 USyncMode	0 Non-CD data 1 CD user channel subcode
0 -	This field is reserved. Reserved.

59.5.3 PhaseConfig Register (SPDIF_SRPC)

Address: 200_4000h base + 8h offset = 200_4008h



SPDIF_SRPC field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23–11 Reserved	This read-only field is reserved and always has the value 0.
10–7 ClkSrc_Sel	Clock source selection, all other settings not shown are reserved: 0000 if (DPLL Locked) SPDIF_RxClk else REF_CLK_32K (XTALOSC) 0001 if (DPLL Locked) SPDIF_RxClk else tx_clk (SPDIF0_CLK_ROOT) 0010 if (DPLL Locked) SPDIF_RxClk else ASRC_EXT_CLK 0011 if (DPLL Locked) SPDIF_RxClk else SPDIF_EXT_CLK 0100 if (DPLL Locked) SPDIF_Rxclk else ESAI_HCKT 0101 REF_CLK_32K (XTALOSC) 0110 tx_clk (SPDIF0_CLK_ROOT) 0111 ASRC_CLK 1000 SPDIF_EXT_CLK 1001 ESAI_HCKT 1010 if (DPLL Locked) SPDIF_RxClk else MLB Clock

Table continues on the next page...

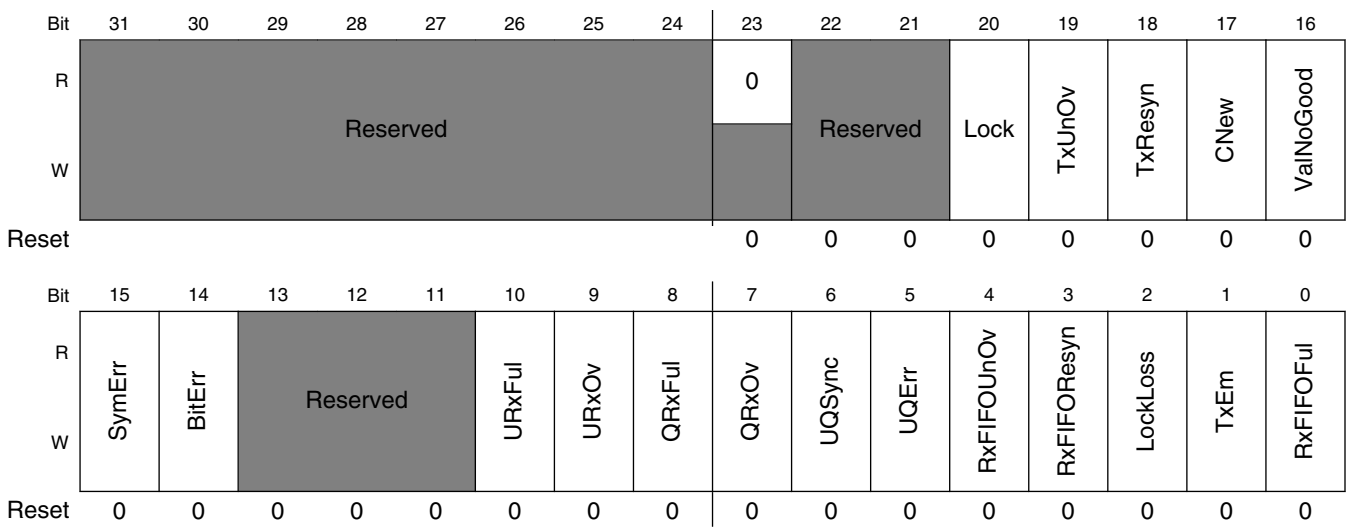
SPDIF_SRPC field descriptions (continued)

Field	Description
	1011 if (DPLL Locked) SPDIF_RxClk else MLB PHY Clock 1100 MLB Clock 1101 MLB PHY Clock
6 LOCK	LOCK bit to show that the internal DPLL is locked, read only
5–3 GainSel	Gain selection: 000 24*(2**10) 001 16*(2**10) 010 12*(2**10) 011 8*(2**10) 100 6*(2**10) 101 4*(2**10) 110 3*(2**10)
Reserved	This read-only field is reserved and always has the value 0.

59.5.4 InterruptEn Register (SPDIF_SIE)

The InterruptEn register (SPDIF_SIE) provides control over the enabling of interrupts.

Address: 200_4000h base + Ch offset = 200_400Ch



SPDIF_SIE field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented.

Table continues on the next page...

SPDIF_SIE field descriptions (continued)

Field	Description
	This field is reserved.
23 Reserved	This read-only field is reserved and always has the value 0.
22–21 -	This field is reserved. Reserved. set to zero.
20 Lock	SPDIF receiver's DPLL is locked
19 TxUnOv	SPDIF Tx FIFO under/overrun
18 TxResyn	SPDIF Tx FIFO resync
17 CNew	SPDIF receive change in value of control channel
16 ValNoGood	SPDIF validity flag no good
15 SymErr	SPDIF receiver found illegal symbol
14 BitErr	SPDIF receiver found parity bit error
13–11 -	This field is reserved. Reserved. set to zero.
10 URxFul	U Channel receive register full, can't be cleared with reg. IntClear. To clear it, read from U Rx reg.
9 URxOv	U Channel receive register overrun
8 QRxFul	Q Channel receive register full, can't be cleared with reg. IntClear. To clear it, read from Q Rx reg.
7 QRxOv	Q Channel receive register overrun
6 UQSync	U/Q Channel sync found
5 UQErr	U/Q Channel framing error
4 RxFIFOUnOv	Rx FIFO underrun/overrun
3 RxFIFOResyn	Rx FIFO resync
2 LockLoss	SPDIF receiver loss of lock
1 TxEm	SPDIF Tx FIFO empty, can't be cleared with reg. IntClear. To clear it, write toTx FIFO.
0 RxFIFOFul	SPDIF Rx FIFO full, can't be cleared with reg. IntClear. To clear it, read from Rx FIFO.

59.5.5 InterruptStat Register (SPDIF_SIS)

The InterruptStat (SPDIF_SIS) register is a read only register that provides the status on interrupt operations.

Address: 200_4000h base + 10h offset = 200_4010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved								0			Lock	TxUnOv	TxResyn	CNew	ValNoGood
W																
Reset									0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SymErr	BitErr	0			URxFul	URxOv	QRxFul	QRxOv	UQSync	UQErr	RxFIFOUnOv	RxFIFOResyn	LockLoss	TxEm	RxFIFOFul
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

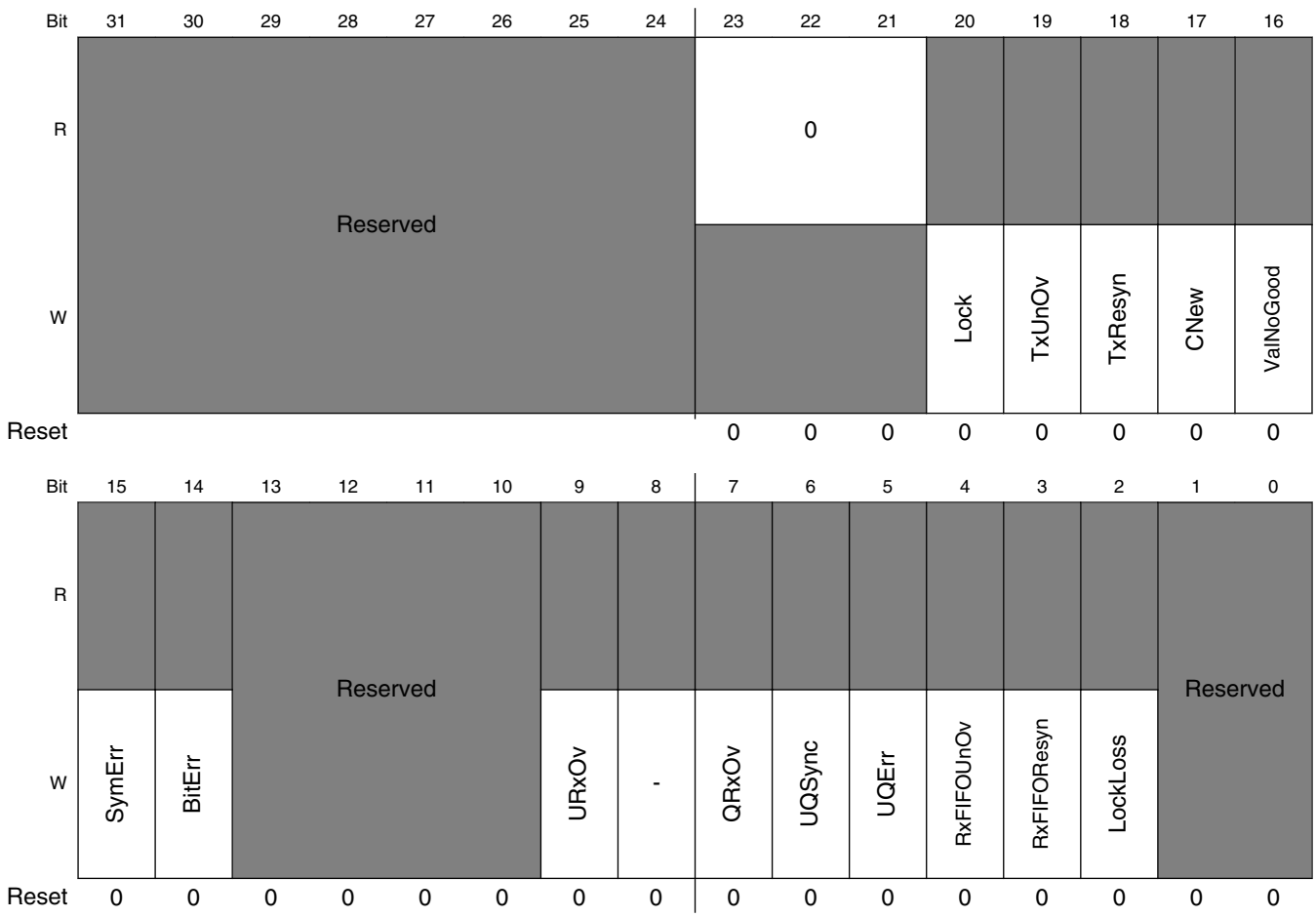
SPDIF_SIS field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23–21 Reserved	This read-only field is reserved and always has the value 0.
20 Lock	SPDIF receiver's DPLL is locked
19 TxUnOv	SPDIF Tx FIFO under/overrun
18 TxResyn	SPDIF Tx FIFO resync
17 CNew	SPDIF receive change in value of control channel
16 ValNoGood	SPDIF validity flag no good
15 SymErr	SPDIF receiver found illegal symbol
14 BitErr	SPDIF receiver found parity bit error
13–11 Reserved	This read-only field is reserved and always has the value 0.
10 URxFul	U Channel receive register full, can't be cleared with reg. IntClear. To clear it, read from U Rx reg.
9 URxOv	U Channel receive register overrun
8 QRxFul	Q Channel receive register full, can't be cleared with reg. IntClear. To clear it, read from Q Rx reg.
7 QRxOv	Q Channel receive register overrun
6 UQSync	U/Q Channel sync found
5 UQErr	U/Q Channel framing error
4 RxFIFOUnOv	Rx FIFO underrun/overrun
3 RxFIFOResyn	Rx FIFO resync
2 LockLoss	SPDIF receiver loss of lock
1 TxEm	SPDIF Tx FIFO empty, can't be cleared with reg. IntClear. To clear it, write toTx FIFO.
0 RxFIFOFul	SPDIF Rx FIFO full, can't be cleared with reg. IntClear. To clear it, read from Rx FIFO.

59.5.6 InterruptClear Register (SPDIF_SIC)

The InterruptClear (SPDIF_SIC) register is a write only register and is used to clear interrupts.

Address: 200_4000h base + 10h offset = 200_4010h



SPDIF_SIC field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23–21 Reserved	This read-only field is reserved and always has the value 0.
20 Lock	SPDIF receiver's DPLL is locked
19 TxUnOv	SPDIF Tx FIFO under/overrun

Table continues on the next page...

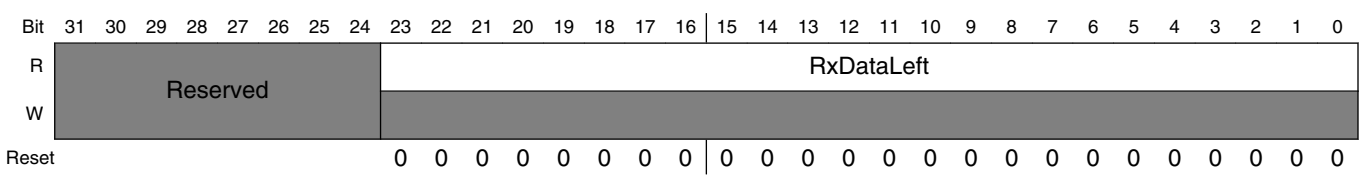
SPDIF_SIC field descriptions (continued)

Field	Description
18 TxResyn	SPDIF Tx FIFO resync
17 CNew	SPDIF receive change in value of control channel
16 ValNoGood	SPDIF validity flag no good
15 SymErr	SPDIF receiver found illegal symbol
14 BitErr	SPDIF receiver found parity bit error
13–10 -	This field is reserved. Reserved.
9 URxOv	U Channel receive register overrun
8 -	Reserved
7 QRxOv	Q Channel receive register overrun
6 UQSync	U/Q Channel sync found
5 UQErr	U/Q Channel framing error
4 RxFIFOUnOv	Rx FIFO underrun/overrun
3 RxFIFOResyn	Rx FIFO resync
2 LockLoss	SPDIF receiver loss of lock
-	This field is reserved. Reserved.

59.5.7 SPDIFRxLeft Register (SPDIF_SRL)

SPDIFRxLeft register is an audio data reception register.

Address: 200_4000h base + 14h offset = 200_4014h



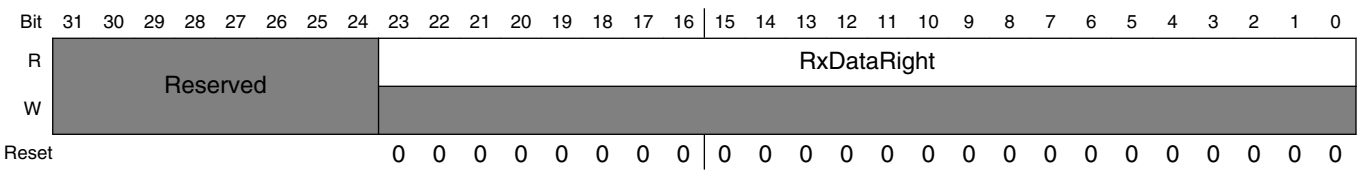
SPDIF_SRL field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
RxDataLeft	Processor receive SPDIF data left

59.5.8 SPDIFRxRight Register (SPDIF_SRR)

SPDIFRxRight register is an audio data reception register.

Address: 200_4000h base + 18h offset = 200_4018h



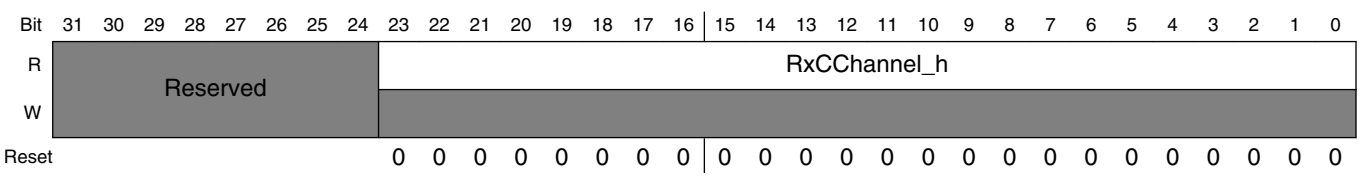
SPDIF_SRR field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
RxDataRight	Processor receive SPDIF data right

59.5.9 SPDIFRxCChannel_h Register (SPDIF_SRCSH)

SPDIFRxCChannel_h register is a channel status reception register.

Address: 200_4000h base + 1Ch offset = 200_401Ch



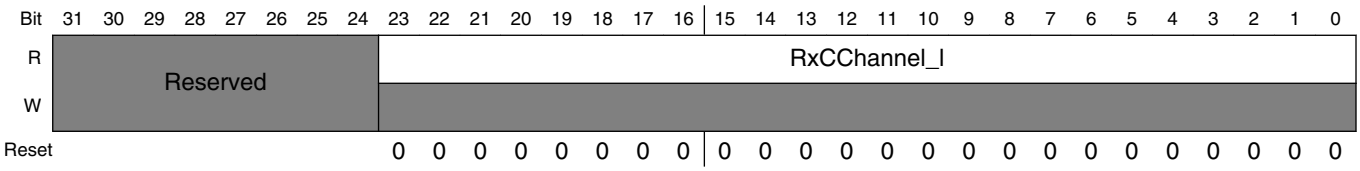
SPDIF_SRCSH field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
RxCChannel_h	SPDIF receive C channel register, contains first 24 bits of C channel without interpretation

59.5.10 SPDIFRxCChannel_l Register (SPDIF_SRCSL)

SPDIFRxCChannel_l register is a channel status reception register.

Address: 200_4000h base + 20h offset = 200_4020h



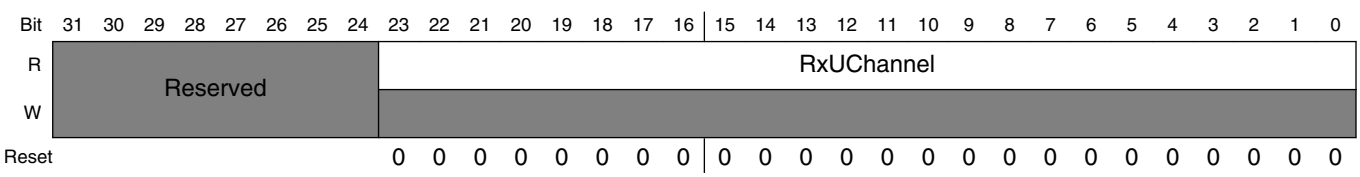
SPDIF_SRCSL field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
RxCChannel_l	SPDIF receive C channel register, contains next 24 bits of C channel without interpretation

59.5.11 UchannelRx Register (SPDIF_SRU)

UChannelRx register is a user bits reception register.

Address: 200_4000h base + 24h offset = 200_4024h



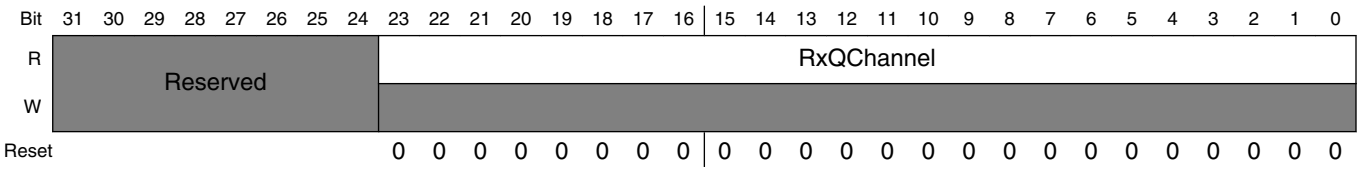
SPDIF_SRU field descriptions

Field	Description
31–24 [unimplemented]	This field is reserved. This is a 24-bit register the upper byte is unimplemented.
RxUChannel	SPDIF receive U channel register, contains next 3 U channel bytes

59.5.12 QchannelRx Register (SPDIF_SRQ)

QChannelRx register is a user bits reception register.

Address: 200_4000h base + 28h offset = 200_4028h



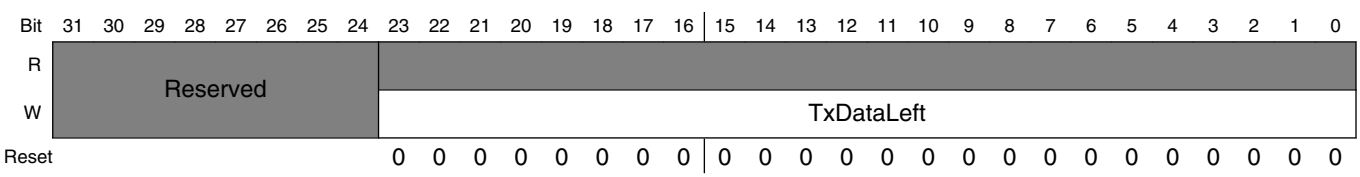
SPDIF_SRQ field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
RxQChannel	SPDIF receive Q channel register, contains next 3 Q channel bytes

59.5.13 SPDIFTxLeft Register (SPDIF_STL)

SPDIFTxLeft register is an audio data transmission register.

Address: 200_4000h base + 2Ch offset = 200_402Ch



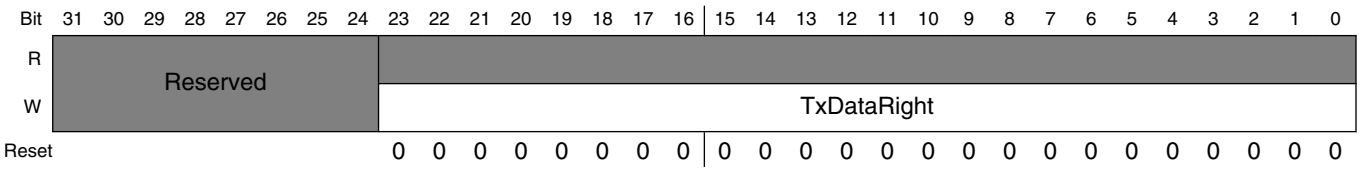
SPDIF_STL field descriptions

Field	Description
31–24 [unimplemented]	This field is reserved. This is a 24-bit register the upper byte is unimplemented.
TxDataLeft	SPDIF transmit left channel data. It is write-only, and always returns zeros when read

59.5.14 SPDIFTxRight Register (SPDIF_STR)

SPDIFTxRight register is an audio data transmission register.

Address: 200_4000h base + 30h offset = 200_4030h



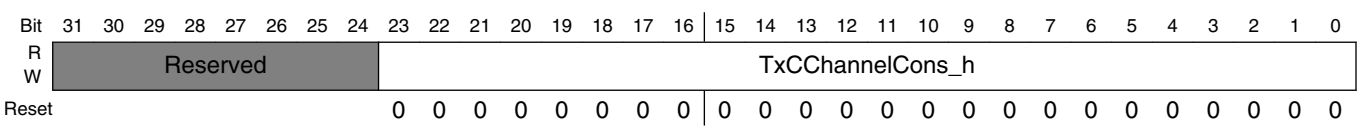
SPDIF_STR field descriptions

Field	Description
31–24 [unimplemented]	This field is reserved. This is a 24-bit register the upper byte is unimplemented.
TxDataRight	SPDIF transmit right channel data. It is write-only, and always returns zeros when read

59.5.15 SPDIFTxCChannelCons_h Register (SPDIF_STCSCH)

SPDIFTxCChannelCons_h register is a channel status transmission register.

Address: 200_4000h base + 34h offset = 200_4034h



SPDIF_STCSCH field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.

Table continues on the next page...

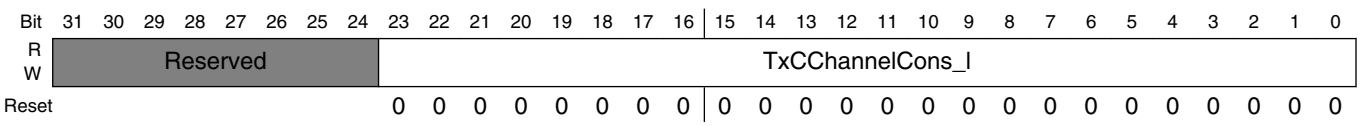
SPDIF_STCSCH field descriptions (continued)

Field	Description
TxCChannelCons_h	SPDIF transmit Cons. C channel data, contains first 24 bits without interpretation. When read, it returns the latest data written by the processor

59.5.16 SPDIFTxCChannelCons_l Register (SPDIF_STCSCL)

SPDIFTxCChannelCons_l register is a channel status transmission register.

Address: 200_4000h base + 38h offset = 200_4038h

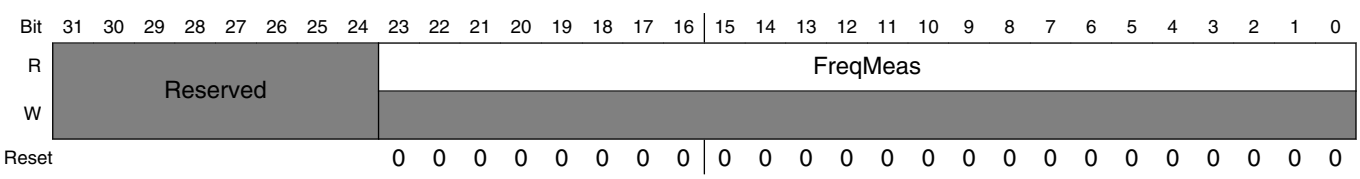


SPDIF_STCSCL field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
TxCChannelCons_l	SPDIF transmit Cons. C channel data, contains next 24 bits without interpretation. When read, it returns the latest data written by the processor

59.5.17 FreqMeas Register (SPDIF_SRFM)

Address: 200_4000h base + 44h offset = 200_4044h



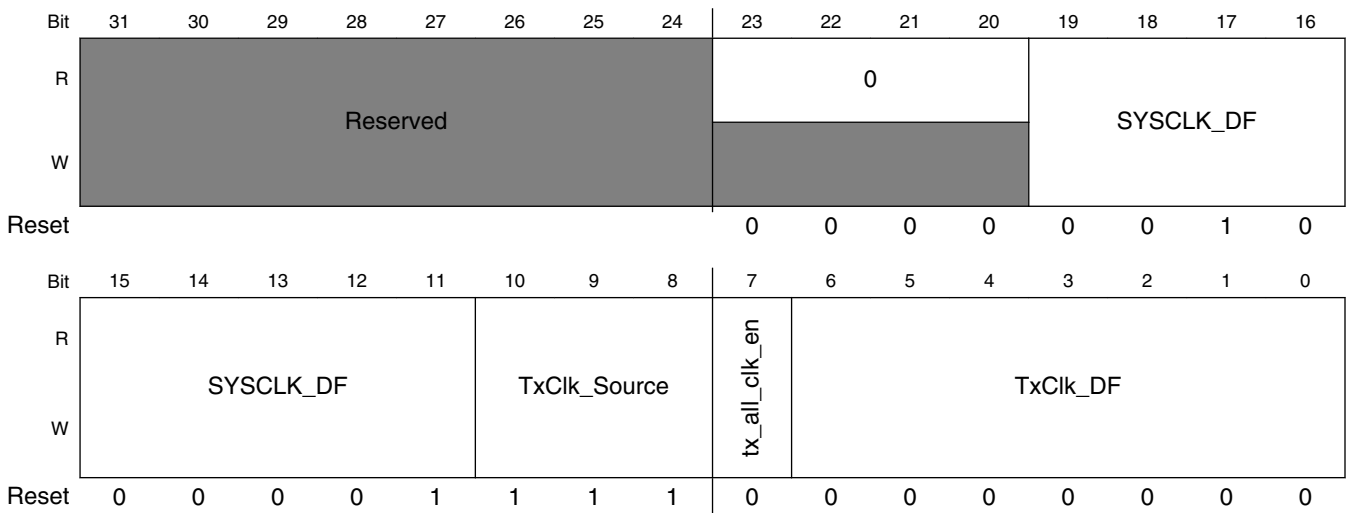
SPDIF_SRFM field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
FreqMeas	Frequency measurement data

59.5.18 SPDIFTxClk Register (SPDIF_STC)

The SPDIFTxClk Control register includes the means to select the transmit clock and frequency division.

Address: 200_4000h base + 50h offset = 200_4050h



SPDIF_STC field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23–20 Reserved	This read-only field is reserved and always has the value 0.
19–11 SYSCLK_DF	system clock divider factor, 2~512. 0 no clock signal 1 divider factor is 2 ... ... 511 divider factor is 512
10–8 TxClk_Source	000 REF_CLK_32K input (XTALOSC 32kHz clock) 001 tx_clk input (from SPDIF0_CLK_ROOT. See CCM.) 010 ASRC_EXT_CLK input 011 SPDIF_EXT_CLK, from pads 100 ESAI_HCKT input 101 ipg_clk input (frequency divided) 110 MLB clock input 111 MLB PHY clock input
7 tx_all_clk_en	Spdif transfer clock enable.When data is going to be transfered, this bit should be set to1.

Table continues on the next page...

SPDIF_STC field descriptions (continued)

Field	Description
	0 disable transfer clock. 1 enable transfer clock.
TxClk_DF	Divider factor (1-128) 0 divider factor is 1 1 divider factor is 2 ... ... 127 divider factor is 128

Chapter 60
System Reset Controller (SRC)

60.1 SRC Overview

The System Reset Controller (SRC) controls the reset and boot operation of the SoC. It is responsible for the generation of all reset signals and boot decoding.

The reset controller determines the source and the type of reset, such as POR, WARM, COLD, and performs the necessary reset qualification and stretching sequences. Based on the type of reset, the reset logic generates the reset sequence for the entire IC. Whenever the chip is powered on, the reset is issued through SRC_ONOFF signal and the entire chip is reset.

60.1.1 Features

The SRC includes the following features.

- Receives and handles the resets from all the reset sources
- Resets the appropriate domains based upon the resets sources and the nature of the reset
- Latches the SRC_BOOT_MODE pins and common configuration signals from the internal fuse

60.2 External Signals

The table found here describes the external signals of SRC.

The following table describes the external signals of SRC:

Table 60-1. SRC External Signals

Signal	Description	Pad	Mode	Direction
SRC_BOOT_CFG00	Boot configuration signal	EIM_DA0	ALT7	I
SRC_BOOT_CFG01	Boot configuration signal	EIM_DA1	ALT7	I
SRC_BOOT_CFG02	Boot configuration signal	EIM_DA2	ALT7	I
SRC_BOOT_CFG03	Boot configuration signal	EIM_DA3	ALT7	I
SRC_BOOT_CFG04	Boot configuration signal	EIM_DA4	ALT7	I
SRC_BOOT_CFG05	Boot configuration signal	EIM_DA5	ALT7	I
SRC_BOOT_CFG06	Boot configuration signal	EIM_DA6	ALT7	I
SRC_BOOT_CFG07	Boot configuration signal	EIM_DA7	ALT7	I
SRC_BOOT_CFG08	Boot configuration signal	EIM_DA8	ALT7	I
SRC_BOOT_CFG09	Boot configuration signal	EIM_DA9	ALT7	I
SRC_BOOT_CFG10	Boot configuration signal	EIM_DA10	ALT7	I
SRC_BOOT_CFG11	Boot configuration signal	EIM_DA11	ALT7	I
SRC_BOOT_CFG12	Boot configuration signal	EIM_DA12	ALT7	I
SRC_BOOT_CFG13	Boot configuration signal	EIM_DA13	ALT7	I
SRC_BOOT_CFG14	Boot configuration signal	EIM_DA14	ALT7	I
SRC_BOOT_CFG15	Boot configuration signal	EIM_DA15	ALT7	I
SRC_BOOT_CFG16	Boot configuration signal	EIM_A16	ALT7	I
SRC_BOOT_CFG17	Boot configuration signal	EIM_A17	ALT7	I
SRC_BOOT_CFG18	Boot configuration signal	EIM_A18	ALT7	I
SRC_BOOT_CFG19	Boot configuration signal	EIM_A19	ALT7	I
SRC_BOOT_CFG20	Boot configuration signal	EIM_A20	ALT7	I
SRC_BOOT_CFG21	Boot configuration signal	EIM_A21	ALT7	I
SRC_BOOT_CFG22	Boot configuration signal	EIM_A22	ALT7	I
SRC_BOOT_CFG23	Boot configuration signal	EIM_A23	ALT7	I
SRC_BOOT_CFG24	Boot configuration signal	EIM_A24	ALT7	I
SRC_BOOT_CFG25	Boot configuration signal	EIM_WAIT	ALT7	I
SRC_BOOT_CFG26	Boot configuration signal	EIM_LBA	ALT7	I
SRC_BOOT_CFG27	Boot configuration signal	EIM_EB0	ALT7	I
SRC_BOOT_CFG28	Boot configuration signal	EIM_EB1	ALT7	I
SRC_BOOT_CFG29	Boot configuration signal	EIM_RW	ALT7	I
SRC_BOOT_CFG30	Boot configuration signal	EIM_EB2	ALT7	I
SRC_BOOT_CFG31	Boot configuration signal	EIM_EB3	ALT7	I
SRC_BOOT_MODE0	Boot mode signal	BOOT_MODE0	No muxing	I
SRC_BOOT_MODE1	Boot mode signal	BOOT_MODE1	No muxing	I
SRC_ONOFF	ONOFF signal	ONOFF	No muxing	I
SRC_POR_B	Power on reset signal	POR_B	No muxing	I

60.3 Clocks

The table found here describes the clock sources for SRC.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 60-2. SRC Clocks

Clock name	Clock Root	Description
ipg_clk	ipg_clk_root	Peripheral clock
ipg_clk_s	ipg_clk_root	Peripheral access clock

60.4 Top-level resets, power-up sequence and external supply integration

Information found here defines chip resets, power-up sequence, and external supply integration.

60.4.1 Reset and Power-up Flow

The chip presumes the following reset and power-up flow:

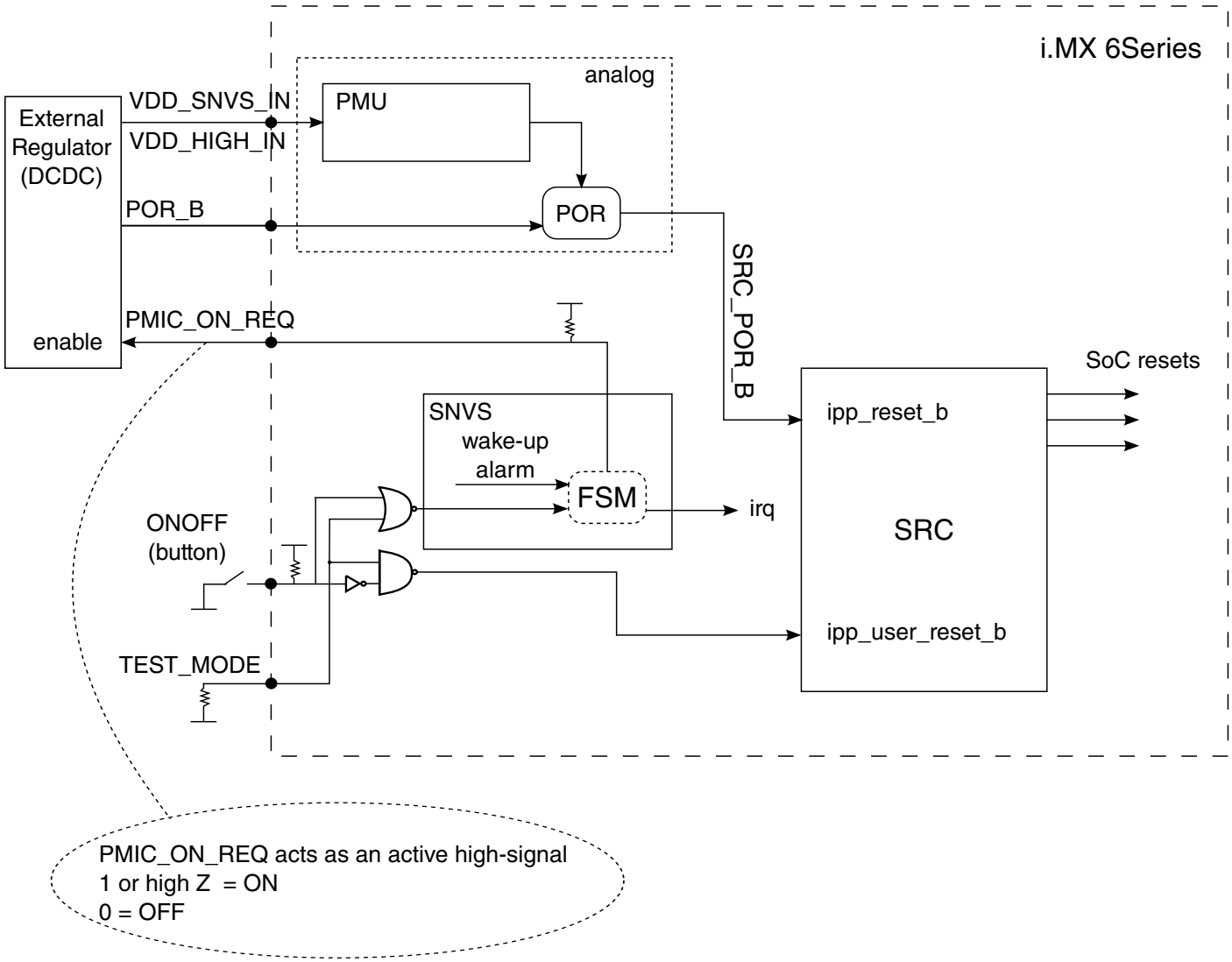


Figure 60-1. Chip reset scheme under PMU control

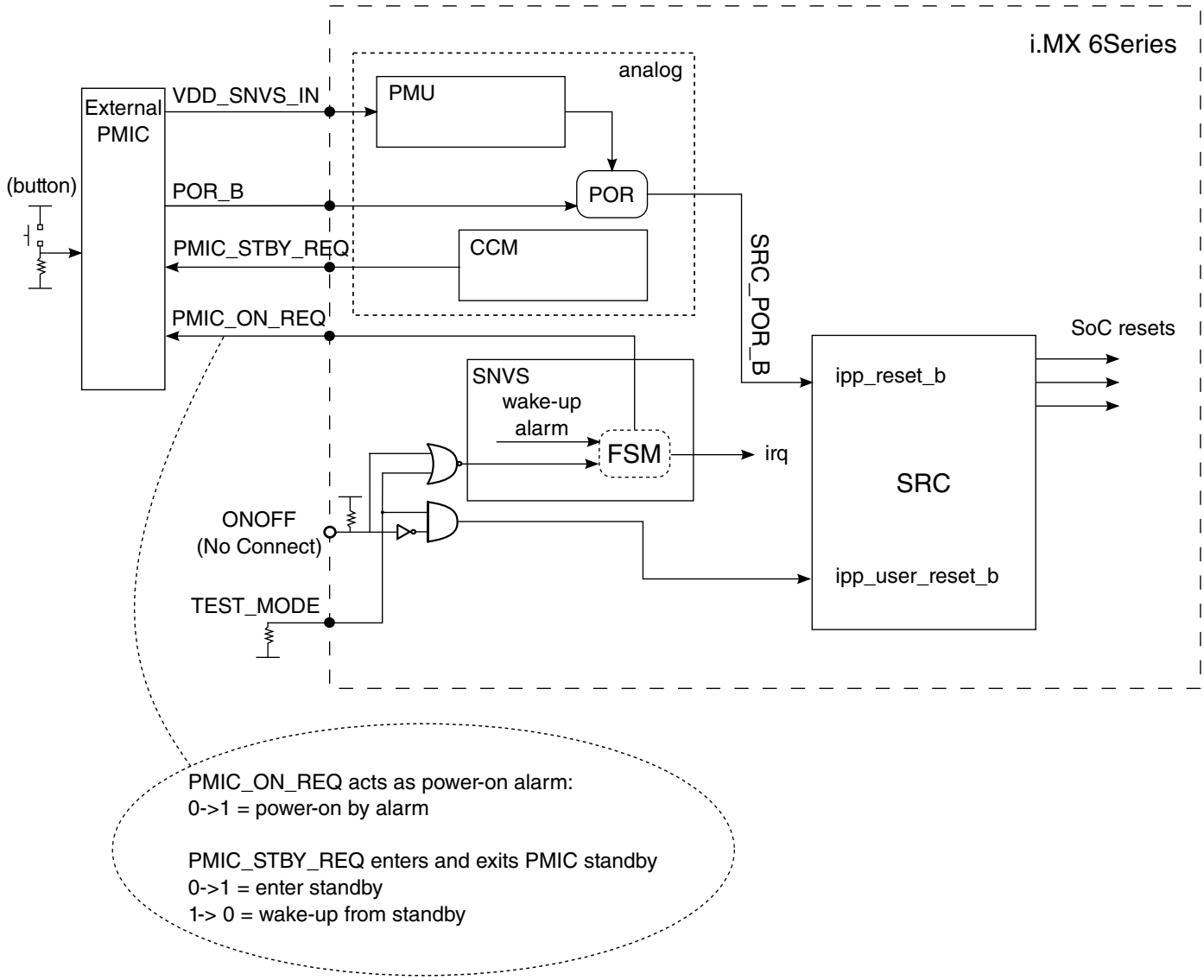


Figure 60-2. Chip reset scheme under external PMIC control

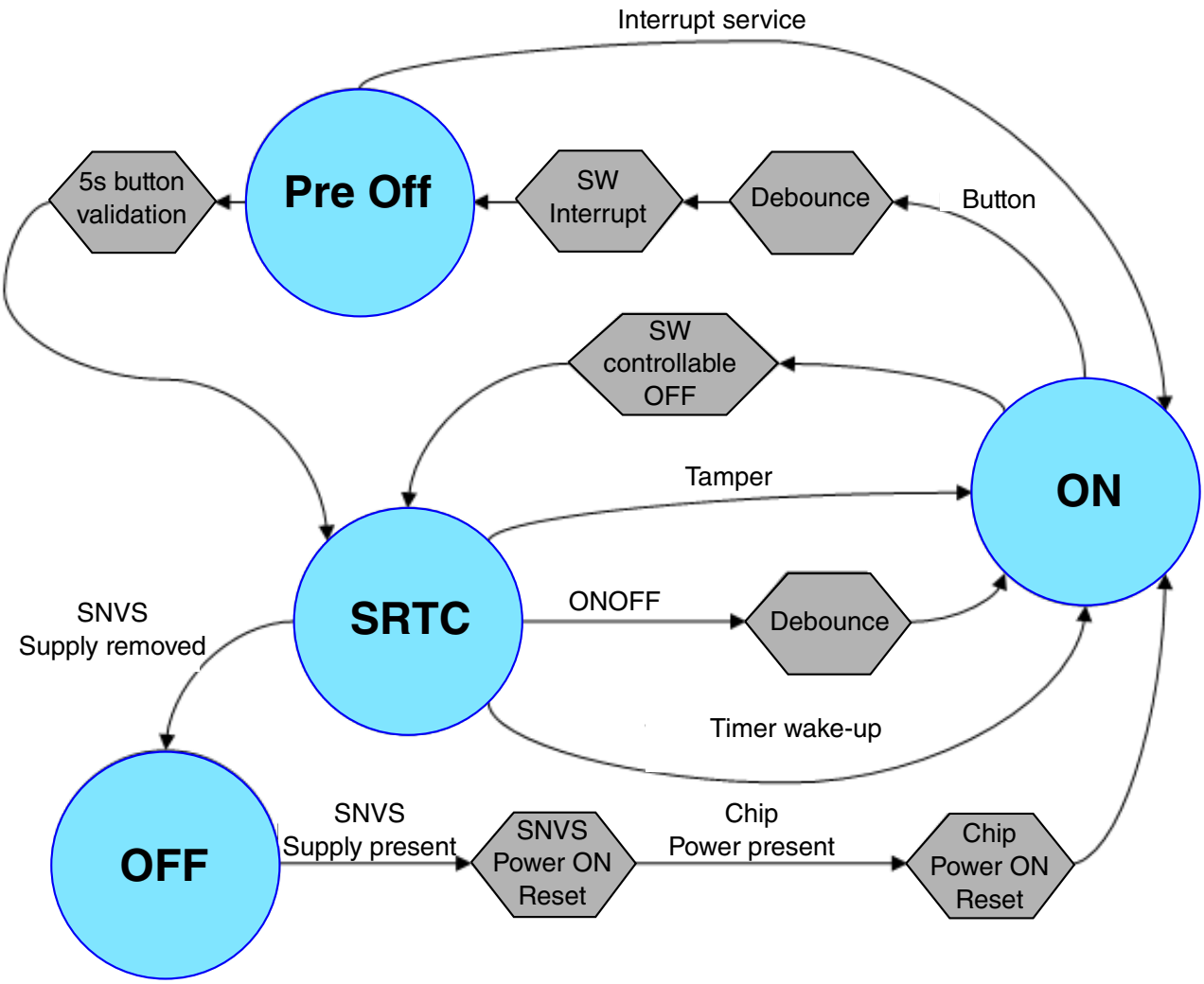


Figure 60-3. Chip on/off state flow diagram

60.4.2 Finite-State Machine (FSM)

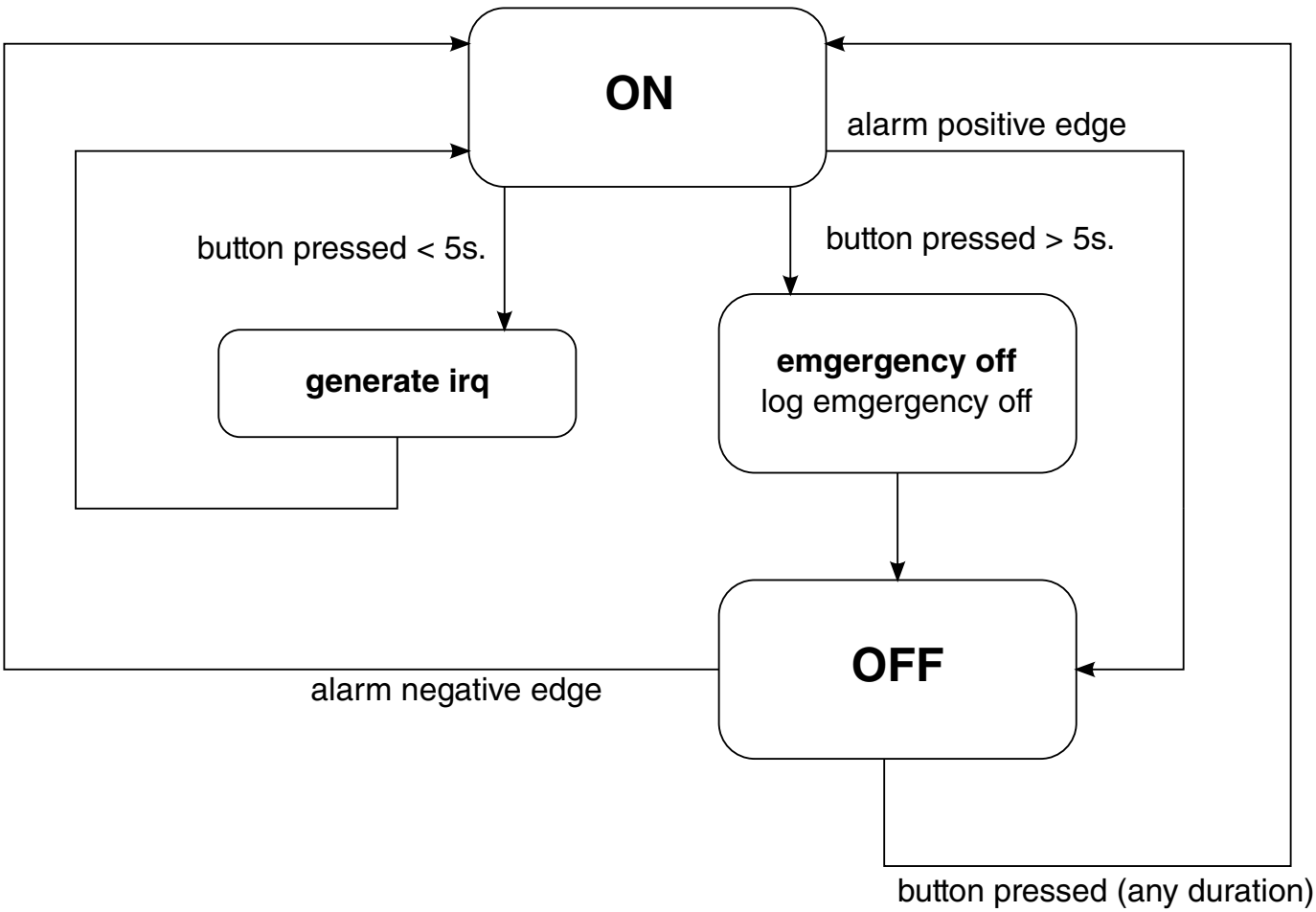


Figure 60-4. FSM

60.4.3 Power mode transitions

Table 60-3. Power mode transitions

Power mode	Configuration with external PMIC	Configuration with internal PMIC
ON, first time	1. Either coin cell or SoC power supply is connected to SNVS. 2. When button is pressed, PMIC powers on.	1. Either coin cell or SoC power supply is connected to SNVS. 2. When button is pressed, 'state' goes ON, PMIC_ON_REQ goes '1'. 3. External regulator is enabled.
Normal ON to OFF, by button	1. Button is pressed for a short duration on the external PMIC. 2. Interrupt request (irq) is sent to SoC from external PMIC.	1. SoC button is pressed for a short duration. 2. Interrupt request (irq) is sent to SoC from FSM. 3. Alarm timer is set up by software routine and started.

Table continues on the next page...

Table 60-3. Power mode transitions (continued)

Power mode	Configuration with external PMIC	Configuration with internal PMIC
	3. SoC is programming PMIC for power off when standby is asserted. 4. In CCM STOP mode, Standby is asserted, PMIC gates SoC supplies.	4. Upon alarm_in assertion to '1', PMIC_ON_REQ goes '0'. 5. External regulator goes OFF.
Emergency ON to OFF, by button	1. Button is pressed for an extended time on the external PMIC. 2. PMIC is powering off.	1. Button is pressed for longer than 5 seconds on the SoC. 2. FSM validates button pressed for 5 seconds. 3. Emergency power off is logged, PMIC_ON_REQ goes '0', alarm_mask goes '1'. 4. External regulator goes OFF.
OFF to ON, by button	1. Button is pressed on the external PMIC. 2. PMIC powers ON.	1. Button is pressed on the SoC. 2. PMIC_ON_REQ goes '1', alarm_mask goes '0'. 3. External regulator powers ON.
OFF to ON, by timer alarm	1. Timer alarm in SNVS is programmed by software before SoC goes OFF. 2. SoC enters OFF mode. 3. Upon timer limit, wake up alarm goes '0'. PMIC_ON_REQ goes '1'. 4. PMIC receives assertion of PMIC_ON_REQ and wakes up.	1. Timer alarm in SNVS is programmed by software before SoC goes OFF. 2. SoC enters OFF mode. 3. Upon timer limit, wake up alarm goes '0'. PMIC_ON_REQ goes '1'. 4. External regulator is enabled by PMIC_ON_REQ = 1.

60.5 Power-On Reset and power sequencing

This module generates an internal POR_B signal that is logically AND'ed with any externally applied SRC_POR_B signal. The internal POR_B signal will be held low until all of the following conditions are met:

- 4ms after the external power supply VDDHIGH_IN is valid
- 1ms after the VDD_SOC_CAP supply is valid

The 4ms and 1ms delays are derived from counting the 32kHz RTC clock cycles; the accuracy depends on the accuracy of the RTC. When the RTC crystal is either absent or in the process of powering up, an internal ring oscillator will be the source of RTC, which is not as accurate as the crystal.

60.5.1 External POR using SRC_POR_B

If the external SRC_POR_B signal is used to control the processor POR, SRC_POR_B must remain low (asserted) until the VDD_ARM_CAP and VDD_SOC_CAP supplies are stable.

60.5.2 Internal POR

If the external SRC_POR_B signal is not used (always held high or left unconnected), the processor defaults to the internal POR function (PMU controls generation of the POR based on the power supplies).

If the internal POR function is used, the following power supply requirements must be met:

- VDD_ARM_IN and VDD_SOC_IN may be supplied from the same source, or
- VDD_SOC_IN can be supplied before VDD_ARM_IN with a maximum delay of 1 ms.

60.6 Functional Description

60.6.1 Reset Control

This section details the reset control of this device.

NOTE

SNVS resets are discussed in the *Multimedia Applications Processor Security Reference Manual for i.MX 6Dual/6Quad and i.MX 6Solo/6DualLite*.

60.6.1.1 Reset inputs and outputs

The reset control logic receives reset requests from all potential reset sources. All the immediate sources of reset are directly passed to the reset stretching block, whereas the resets requiring qualification are passed on to the reset qualification logic before they are sent to the reset stretching block.

All reset inputs and outputs are described in the following figure:

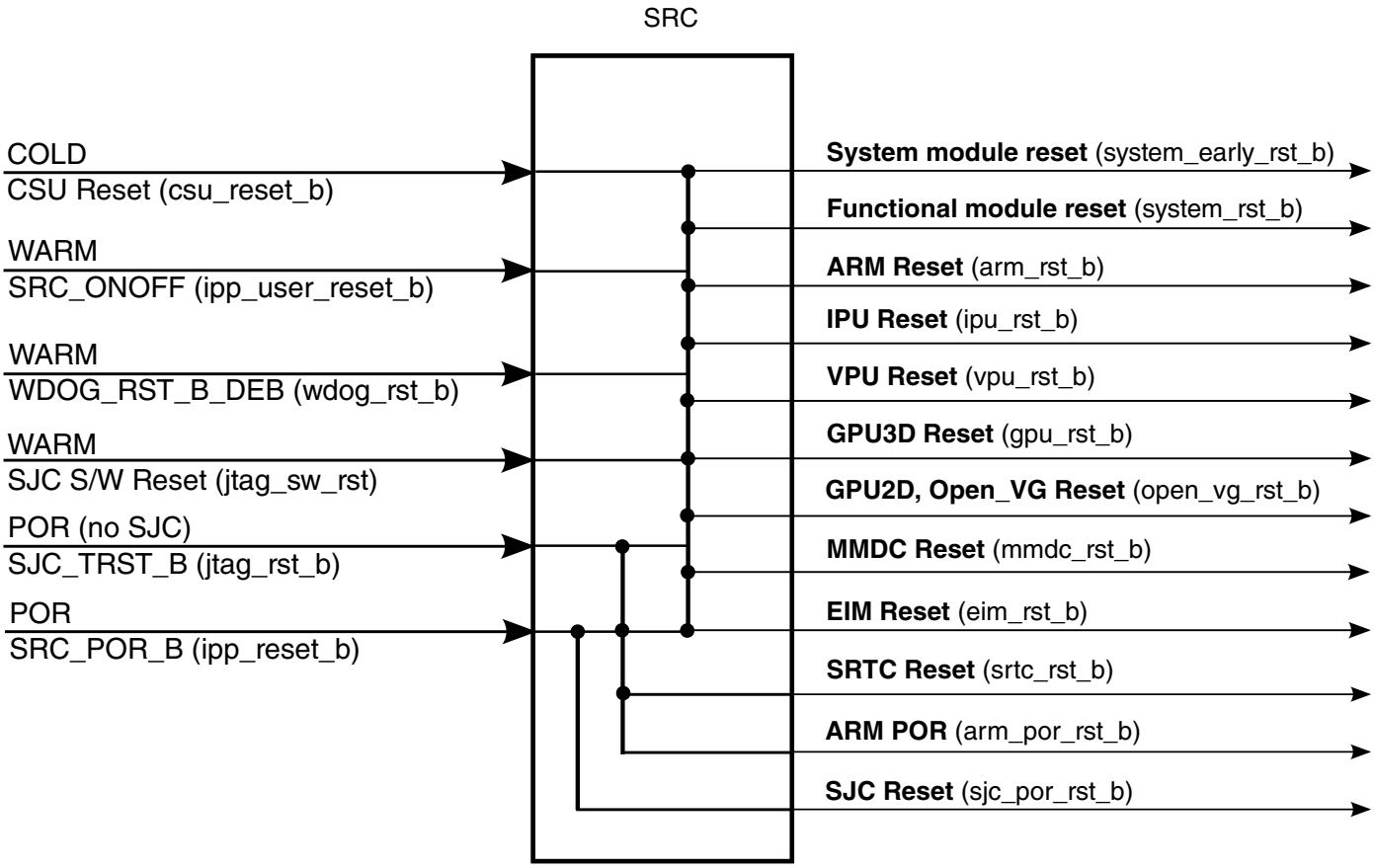


Figure 60-5. SRC inputs and outputs

The reset types and modules they affect are shown in Table 60-4. As there is no chip POR, the POR_B is used to reset the entire chip including test logic and JTAG modules.

NOTE

All resets are expected to be active low except jtag_sw_rst.

Table 60-4. SRC reset functionality

SoC Modules	POR	COLD	WARM
System modules (PLLs, fuses, etc)	yes	yes	yes
Functional modules	yes	yes	yes
ARM	yes	yes	yes
ARM SoC	yes	yes	yes
IPU	yes	yes	yes
VPU	yes	yes	yes
GPU3D	yes	yes	yes
MMDC	yes	yes	yes
ARM POR	yes	no	no
ARM debug	yes	no	no

Table continues on the next page...

Table 60-4. SRC reset functionality (continued)

SoC Modules	POR	COLD	WARM
SJC	yes	no	no
SRTC	yes	no	no

The reset priorities are POR (strongest), COLD and WARM (weakest). If a stronger reset is asserted during the sequence of a weaker reset, then the weaker sequence will be overridden, and the stronger reset sequence will commence. There is no priority within a reset type (POR, etc). If a reset is asserted during the reset sequence of the same type, the reset sequence will be interrupted and restarted.

The following lists the functionality of each of these reset outputs:

- system_early_rst_b - Resets the system modules that need to start first as CCM, OCOTP_CTRL, FUSEBOX, MMDC, etc.
- system_rst_b - Resets functional modules
- arm_rst_b - Resets ARM module (on regular system reset)
- ipu1_rst_b - Resets IPU1 module (on regular system reset)
- ipu2_rst_b - Resets IPU2 module (on regular system reset)
- vpu_rst_b - Resets VPU module (on regular system reset)
- gpu_rst_b - Resets GPU3D module (on regular system reset)
- open_vg_rst_b - Resets Open_VG and GPU2D modules (on regular system reset)
- mmdc_rst_b - Resets MMDC
- arm_por_rst_b - Resets ARM por input
- arm_soc_rst_b - Reset for ARM SOC
- arm_dbg_rst_b - Reset debug logic of ARM
- test_logic_rst_b - Reset test logic (IOMUXC, DAP)
- sjc_por_rst_b - Reset to SJC
- srtc_rst_b - Resets SRTC

NOTE

It is assumed that each reset source will deassert after its assertion, either due to reset generated to the system from SRC, or by negation of the reset source (if it came from an external source to the chip). In the latter case, the reset source is assumed to be held for at least 2 XTALI clocks so it can be sampled by SRC.

60.6.1.2 Reset Handling

60.6.1.2.1 Reset Qualification

The reset qualification logic qualifies the reset source before sending it out to the chip as a valid reset. WARM resets are in this category. All remaining reset sources are immediate resets and are acknowledged by the reset circuitry the moment they are asserted.

WARM resets are not immediate resets. WARM resets do reset the CCM, the source of MMDC clock. So, if a WARM reset were to immediately reset the CCM, then the MMDC clock would be shut off and this may cause the MMDC data to be lost. During normal mode of operation of the chip, the protocol that is followed before shutting off the MMDC clock is that an mmdc_dvfs_req signal is sent to MMDC and only after the MMDC sends an acknowledge signal, mmdc_dvfs_ack, is the clock to the MMDC gated off.

However, the implication here is that a valid WARM reset source condition will not be able to cause a chip reset until the MMDC sends the acknowledge signal (mmdc_dvfs_ack). For example, a JTAG reset event has occurred but the JTAG reset will not be serviced until the mmdc_dvfs_ack signal is received. So, essentially all WARM reset sources depend on the MMDC providing the mmdc_dvfs_ack acknowledge signal before the reset is performed. When the MMDC is not used, mmdc_dvfs_ack is defaulted high.

The occurrence of WARM reset results in the assertion of warm_reset signal before the system resets are asserted to indicate to the MMDC that the reset occurred is a WARM reset.

A reset source is updated in the Reset Status Register (SRC_SRSR) when it becomes valid, provided it is asserted for the minimum amount of time after asserting. So, all immediate resets are immediately updated in the Status Register (SRC_SRSR). WARM resets would be updated when the mmdc_dvfs_ack signal is received from the MMDC.

Once the reset is qualified, depending on the source of the reset, internal resets are asserted appropriately.

60.6.1.2.2 Reset Sequence and De-Assertion

The SRC_ONOFF will assert immediately after any reset source is recognized (except for the case of WARM reset when MMDC needs to answer mmdc_dvfs_ack first). After all of the reset sources are released, the SRC will start the following set of events depending on the type of reset that occurred.

60.6.1.2.3 POR (SRC_POR_B)

SRC_POR_B is an external reset signal. When the chip is powered up, the reset signal is passed through the POR_B pin indicating power-up sequence. The SRC resets the entire chip including the JTAG (SJC) module. All SRC registers will be reset during the POR sequence.

As soon as SRC_POR_B occurs, all resets are asserted and the entire chip is reset by SRC. The SRC_POR_B is stretched for 2 XTALI cycles and the stretching sequence takes place after 2 XTALI clocks of POR_B pin deassertion.

The sjc_por_rst_b and srtc_rst_b signals are deasserted together with SRC_POR_B signal. Those outputs are also deasserted after the stretching of SRC_POR_B has deasserted.

Once the above resets deassert, system_early_rst_b reset is deasserted after 2 XTALI clocks. The system_early_rst_b is used for the CCM and PLL-IPs to start generating PLL clock ouputs and the system root clocks.

When the system root clocks are ready, the CCM will assert system_clk_ready signal. This signal is generated during the start sequence in the CCM and it involves the preparation of the PLLs to generate clock roots for functional operation.

SRC then enables OCOTP_CTRL and fusebox clocks, so that fuses can be loaded to OCOTP_CTRL.

- SRC will prepare the boot information and then de assert mmdc_rst_b.
- SRC will wait 8 ipg clock cycles, and then SRC will enable MMDC clocks.
- SRC will wait 8 XTALI clocks to allow MMDC to generate fixed external clock to external memory SDRAM.
- SRC will enable VPU and GPU clocks.
- After 8 ipg cycles, SRC will disable VPU, Open_VG and GPU clocks.
- After 8 ipg cycles, resets to all modules will be de-asserted (system_rst_b, vpu_rst_b, gpu_rst_b, open_vg_rst_b, ipu_rst_b).
- After 8 ipg cycles, system clocks will be enabled (en_system_clk).

60.6.1.2.4 COLD RESET

The sequence is similar to SRC_POR_B except the memory repair operation is not performed.

Once the reset source deasserts, system_early_rst_b reset is deasserted after at least 2 XTALI clocks. The system_early_rst_b is used for the CCM and PLL-IPs to start generating PLL clock outputs and the system root clocks.

Once the system root clocks are ready, the CCM will assert system_clk_ready signal. This signal is generated during the start sequence in the CCM and it involves the preparation of the PLLs to generate clock roots for functional operation. See CCM for more information.

Once system_clk_ready arrives at the SRC, it will enable OCOTP_CTRL and fusebox clocks, so that fuses can be loaded to OCOTP_CTRL. OCOTP_CTRL will notify with iim_ready_flag once the fusebox loading finishes.

- SRC will prepare the boot information and then deassert mmdc_rst_b.
- SRC will wait 8 ipg clock cycles, and then SRC will enable MMDC clocks.
- SRC will wait 8 XTALI clocks to allow MMDC to generate fixed external clock to external memory SDRAM.
- SRC will enable VPU and GPU clocks so that reset will penetrate this module.
- After 8 ipg cycles, SRC will disable VPU, Open_VG and GPU clocks.
- After 8 ipg cycles resets to all modules will be deasserted (system_rst_b, vpu_rst_b, gpu_rst_b, open_vg_rst_b, ipu_rst_b).
- After 8 ipg cycles, system clocks will be enabled (en_system_clk).

60.6.1.2.5 WARM RESET

WARM reset will be enabled only if SRC_SCR[warm_reset_enable] bit is programmed. Otherwise, all WARM reset sources will generate a COLD reset. This bit will be reset only by a POR.

A WARM reset is similar to a COLD reset except that before the reset is sent, a signal to MMDC is asserted mmdc_dvfs_req (generates DVFS assertion to MMDC) to request to prepare MMDC to a WARM reset, finishing the transactions placing MMDC in self-refresh. Another signal will be asserted to MMDC (warm_reset) that will wrap the WARM reset sequence and will notify MMDC that a WARM reset is in process.

One of the sources of the WARM reset is SRC_ONOFF. If this is the case, it is qualified for 4 XTALI edges. The WARM reset is initiated immediately after 4 XTALI edges. The system does not come out of reset until the the SRC_ONOFF is released.

In case the handshake mechanism with MMDC is stuck, meaning that no mmdc_dvfs_ack is recieved, COLD reset will be generated after a number of XTALI clocks. The number of XTALI clocks is defined in register the SRC_SCR[warm_rst_bypass_count] bitfield (default of this bitfield is 16 XTALI counts.)

The following is a basic description of the WARM reset sequence:

1. ARM sets SRC_SCR[warm_reset_enable] bit to enable the WARM Reset functionality. If this bit is not set, all WARM reset sources will result in COLD reset.
2. Assertion of one of the WARM reset sources.

3. The reset source is qualified in the SRC.
4. If mmdc_dvfs_ack signal is low, then SRC triggers the MMDC to switch to self-refresh mode using mmdc_dvfs_req signal. This is done through the CCM to combine with the DVFS sent from the CCM in case of frequency change of MMDC.
5. Wait for mmdc_dvfs_ack signal from the MMDC. If no ack is recieved during warm_rst_bypass_count number of XTALI clocks, COLD reset will be generated.
6. Assert warm_reset signal to MMDC.
7. SRC asserts system resets

The deassertion sequence is exactly the same as in the Cold Reset except waiting for 8 XTALIs for MMDC to generate fixed external clock to external memory MMDC. This stage is not needed in WARM reset since MMDC is held in self-refresh in WARM reset and there is no need to reconfigure it when exiting WARM reset.

WARM BOOT

Software can save any needed information in the memory before initiating a WARM reset. In this case, software will set SRC_SRSR[warm_boot] bit before initiating WARM reset. After the system returns to run mode, the warm_boot bit will still be set, indicating the software that data was saved in memory and can be reused.

NOTE

mmdc_dvfs_req and acknowledge during WARM reset can be masked in the CCM by configuration of register CCDR[17:16].

60.6.2 Parallel Reset Requests

SRC will follow the following rules in the case of parallel reset requests:

1. The order of strength of resets is POR - strongest, cold - medium, warm - weakest.
2. If a stronger reset is asserted during weaker reset sequence, then the stronger reset will take over and the stronger reset process will commence. The following cases fall into this category:
   - POR reset request in the middle of cold or warm reset process - the cold or warm reset process will be stopped and the POR sequence will start.
   - COLD reset request in the middle of warm reset process - the warm reset process will be stopped and the cold sequence will start.
3. If a weaker reset is asserted during stronger reset sequence, then the stronger reset sequence will continue without interference. If at the end of the stronger reset process the weaker request is still asserted then the weaker sequence will commence. The following cases fall into this category:

- COLD or WARM reset requests in the middle of POR reset process - the POR process will continue without interference.
- WARM reset request in the middle of COLD reset process - the COLD process will continue without interference.

4. If a similar reset request is asserted during the process of reset handling, then the process of reset handling will start over (with the same process). The following cases fall into this category:
   - POR reset request in the middle of POR reset process - the POR process will start over.
   - COLD reset request in the middle of COLD reset process - the COLD process will start over.

There is one exception to this category: WARM reset request in the middle of WARM reset process. In this case, the new WARM reset process cannot restart because MMDC is reset and there can't be a handshake with MMDC if it is reset. In this case the first WARM reset will continue, and only if the second WARM reset is still asserted after the first one has finished, the WARM sequence will start again.

60.6.3 Boot Mode Control

60.6.3.1 BOOT_MODE Pin Latching

The exact boot sequence is controlled by the values of the BOOT_MODE pins on this device.

The value of the BOOT_MODE pins will be latched after the OCOTP_CTRL asserts the fuse read completion flag. After latching, the values of the BOOT_MODE pins are used to determine the booting options of the core as described in the SRC_SBMRx registers.

The boot mode general purpose bits can be provided to the SRC from either e-fuses or GPIO signals. The gpio_bt_sel e-fuse defines the source to be used to derive the boot information. When gpio_bt_sel is set, e-fuses are used. When cleared, GPIO signals are used.

The boot information is provided in SRC_SBMR1 register. The figure below shows the selection of boot mode information.

NOTE

BOOT_MODE[1:0] inputs of SRC are connected to BOOT_MODE[1:0] pins.

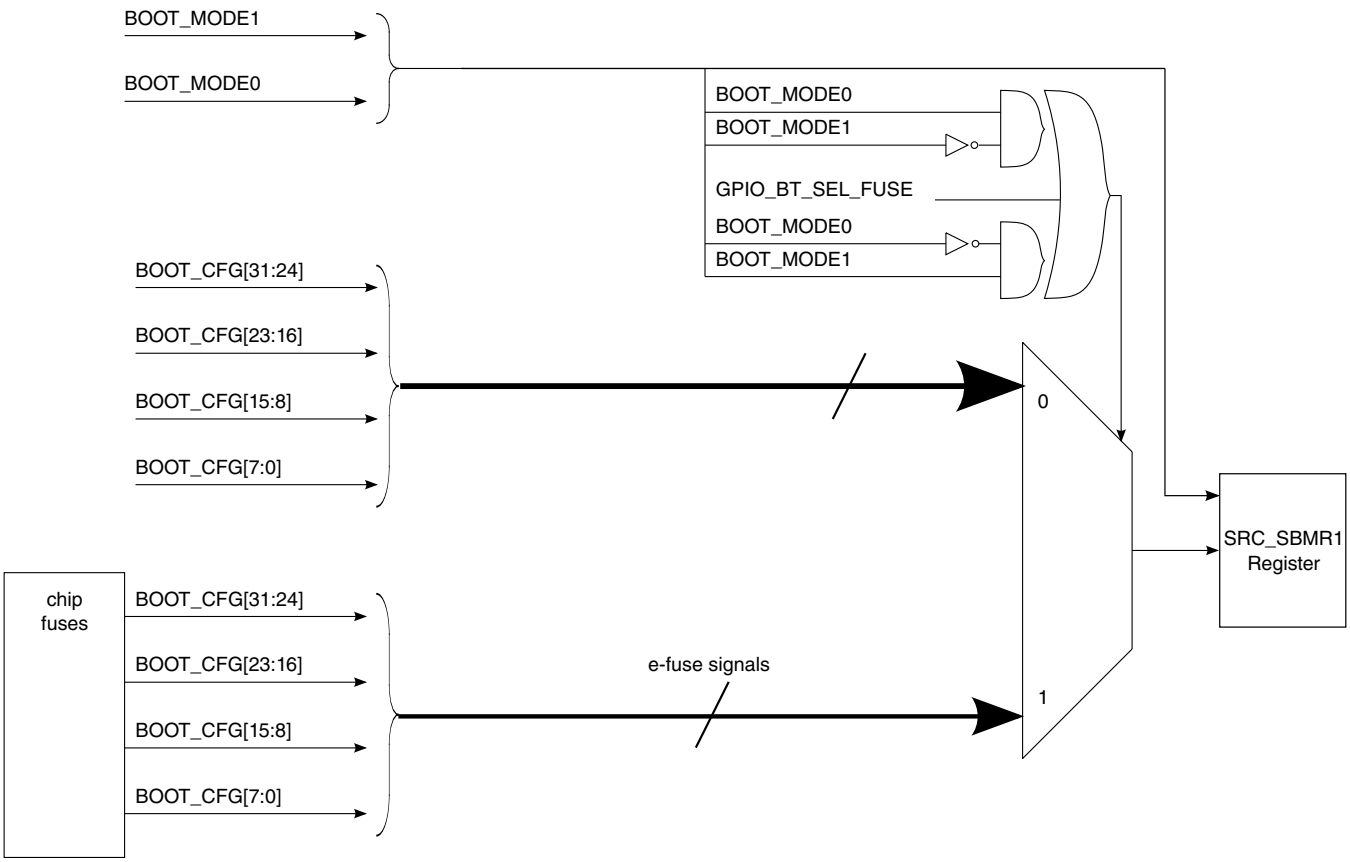


Figure 60-6. Boot mode information

60.7 SRC Memory Map/Register Definition

SRC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20D_8000	SRC Control Register (SRC_SCR)	32	R/W	0000_0521h	60.7.1/5070
20D_8004	SRC Boot Mode Register 1 (SRC_SBMR1)	32	R	0000_0000h	60.7.2/5074
20D_8008	SRC Reset Status Register (SRC_SRSR)	32	R/W	0000_0001h	60.7.3/5075
20D_8014	SRC Interrupt Status Register (SRC_SISR)	32	R	0000_0000h	60.7.4/5077
20D_8018	SRC Interrupt Mask Register (SRC_SIMR)	32	R/W	0000_001Fh	60.7.5/5079
20D_801C	SRC Boot Mode Register 2 (SRC_SBMR2)	32	R	0000_0000h	60.7.6/5080
20D_8020	SRC General Purpose Register 1 (SRC_GPR1)	32	R/W	0000_0000h	60.7.7/5081
20D_8024	SRC General Purpose Register 2 (SRC_GPR2)	32	R/W	0000_0000h	60.7.8/5081
20D_8028	SRC General Purpose Register 3 (SRC_GPR3)	32	R/W	0000_0000h	60.7.9/5082

Table continues on the next page...

SRC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20D_802C	SRC General Purpose Register 4 (SRC_GPR4)	32	R/W	0000_0000h	60.7.10/ 5082
20D_8030	SRC General Purpose Register 5 (SRC_GPR5)	32	R/W	0000_0000h	60.7.11/ 5082
20D_8034	SRC General Purpose Register 6 (SRC_GPR6)	32	R/W	0000_0000h	60.7.12/ 5083
20D_8038	SRC General Purpose Register 7 (SRC_GPR7)	32	R/W	0000_0000h	60.7.13/ 5083
20D_803C	SRC General Purpose Register 8 (SRC_GPR8)	32	R/W	0000_0000h	60.7.14/ 5083
20D_8040	SRC General Purpose Register 9 (SRC_GPR9)	32	R/W	0000_0000h	60.7.15/ 5084
20D_8044	SRC General Purpose Register 10 (SRC_GPR10)	32	R/W	0000_0000h	60.7.16/ 5085

60.7.1 SRC Control Register (SRC_SCR)

The Reset control register (SCR), contains bits that control operation of the reset controller.

Address: 20D_8000h base + 0h offset = 20D_8000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0						dbg_rst_msk_pg	core3_enable	core2_enable	core1_enable	cores_dbg_rst	core3_dbg_rst	core2_dbg_rst	core1_dbg_rst	core0_dbg_rst	core3_rst
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	core2_rst	core1_rst	core0_rst	sw_ipu2_rst	eim_rst	mask_wdog_rst				warm_rst_ bypass_ count		sw_open_vg_rst	sw_ipu1_rst	sw_ vpu_ rst	sw_ gpu_ rst	warm_reset_ enable
W																
Reset	0	0	0	0	0	1	0	1	0	0	1	0	0	0	0	1

SRC_SCR field descriptions

Field	Description
31–26 Reserved	This read-only field is reserved and always has the value 0.
25 dbg_rst_msk_pg	Do not assert debug resets after power gating event of core 0 do not mask core debug resets (debug resets will be asserted after power gating event) 1 mask core debug resets (debug resets won't be asserted after power gating event)
24 core3_enable	core3 enable. **NOTE:** core0 cannot be disabled 0 core3 is disabled 1 core3 is enabled
23 core2_enable	core2 enable. **NOTE:** core0 cannot be disabled 0 core2 is disabled 1 core2 is enabled
22 core1_enable	core1 enable. **NOTE:** core0 cannot be disabled 0 core1 is disabled 1 core1 is enabled
21 cores_dbg_rst	Software reset for debug of arm platform only. **NOTE:** This is a self clearing bit. Once it is set to 1, the reset process will begin, and once it finishes, this bit will be self cleared. 0 do not assert arm platform debug reset 1 assert arm platform debug reset
20 core3_dbg_rst	Software reset for core3 debug only. **NOTE:** This is a self clearing bit. Once it is set to 1, the reset process will begin, and once it finishes, this bit will be self cleared. 0 do not assert core3 debug reset 1 assert core3 debug reset
19 core2_dbg_rst	Software reset for core2 debug only. **NOTE:** This is a self clearing bit. Once it is set to 1, the reset process will begin, and once it finishes, this bit will be self cleared. 0 do not assert core2 debug reset 1 assert core2 debug reset
18 core1_dbg_rst	Software reset for core1 debug only. **NOTE:** This is a self clearing bit. Once it is set to 1, the reset process will begin, and once it finishes, this bit will be self cleared. 0 do not assert core1 debug reset 1 assert core1 debug reset

Table continues on the next page...

SRC_SCR field descriptions (continued)

Field	Description
17 core0_dbg_rst	Software reset for core0 debug only. **NOTE:** This is a self clearing bit. Once it is set to 1, the reset process will begin, and once it finishes, this bit will be self cleared. 0 do not assert core0 debug reset 1 assert core0 debug reset
16 core3_rst	Software reset for core3 only. **NOTE:** This is a self clearing bit. Once it is set to 1, the reset process will begin, and once it finishes, this bit will be self cleared. 0 do not assert core3 reset 1 assert core3 reset
15 core2_rst	Software reset for core2 only. **NOTE:** This is a self clearing bit. Once it is set to 1, the reset process will begin, and once it finishes, this bit will be self cleared. 0 do not assert core2 reset 1 assert core2 reset
14 core1_rst	Software reset for core1 only. **NOTE:** This is a self clearing bit. Once it is set to 1, the reset process will begin, and once it finishes, this bit will be self cleared. 0 do not assert core1 reset 1 assert core1 reset
13 core0_rst	Software reset for core0 only. **NOTE:** This is a self clearing bit. Once it is set to 1, the reset process will begin, and once it finishes, this bit will be self cleared. 0 do not assert core0 reset 1 assert core0 reset
12 sw_ipu2_rst	Software reset for ipu2 **NOTE:** This is a self clearing bit. Once it is set to 1, the reset process will begin, and once it finishes, this bit will be self cleared. Software can determine that the reset has finished once this bit is cleared. Software can also configure SRC to generate interrupt once the software has finished. Please refer to SRC_SISR register for details. 0 do not assert ipu2 reset 1 assert ipu2 reset
11 eim_rst	EIM reset is needed in order to reconfigure the eim chip select. The software reset bit must de-asserted. The eim chip select configuration should be updated. The software bit must be re-asserted since this is not self-refresh.
10–7 mask_wdog_rst	Mask wdog_rst_b source. If these 4 bits are coded from A to 5 then, the wdog_rst_b input to SRC will be masked and the wdog_rst_b will not create a reset to the chip.

Table continues on the next page...

SRC_SCR field descriptions (continued)

Field	Description
	NOTE: During the time the WDOG event is masked using SRC logic, it is likely that the WDOG Reset Status Register (WRSR) bit 1 (which indicates a WDOG timeout event) will get asserted. software / OS developer must prepare for this case. Re-enabling the WDOG is possible, by un-masking it in SRC, though it must be preceded by servicing the WDOG. However, for the case that the event has been asserted, the status bit (WRSR bit-1) will remain asserted, regardless of servicing the WDOG module. (Hardware reset is the only way to cause the de-assertion of that bit). any other code will be coded to 1010 i.e. wdog_rst_b is not masked 0101 wdog_rst_b is masked 1010 wdog_rst_b is not masked (default)
6–5 warm_rst_ bypass_count	Defines the XTALI cycles to count before bypassing the MMDC acknowledge for WARM reset. If the MMDC acknowledge will not be asserted before this counter has elapsed, then a COLD reset will be initiated. 00 Counter not to be used - system will wait until MMDC acknowledge until it is asserted. 01 Wait 16 XTALI cycles before changing WARM reset to a COLD reset. 10 Wait 32 XTALI cycles before changing WARM reset to a COLD reset. 11 Wait 64 XTALI cycles before changing WARM reset to a COLD reset
4 sw_open_vg_rst	Software reset for open_vg **NOTE:** This is a self clearing bit. Once it is set to 1, the reset process will begin, and once it finishes, this bit will be self cleared. Software can determine that the reset has finished once this bit is cleared. Software can also configure SRC to generate interrupt once the software has finished. Please refer to SRC_SISR register for details. **NOTE:** The reset process will involve 8 open_vg cycles before negating the open_vg reset, to allow reset assertion to propagate into open_vg. 0 do not assert open_vg reset 1 assert open_vg reset
3 sw_ipu1_rst	Software reset for IPU1 **NOTE:** This is a self clearing bit. Once it is set to 1, the reset process will begin, and once it finishes, this bit will be self cleared. Software can determine that the reset has finished once this bit is cleared. Software can also configure SRC to generate interrupt once the software has finished. Please refer to SRC_SISR register for details. 0 do not assert IPU1 reset 1 assert IPU1 reset
2 sw_vpu_rst	Software reset for VPU **NOTE:** This is a self clearing bit. Once it is set to 1, the reset process will begin, and once it finishes, this bit will be self cleared. Software can determine that the reset has finished once this bit is cleared. Software can also configure SRC to generate interrupt once the software has finished. Please refer to SRC_SISR register for details. **NOTE:** The reset process will involve 8 VPU cycles before negating the VPU reset, to allow reset assertion to propagate into VPU. 0 do not assert VPU reset 1 assert VPU reset

Table continues on the next page...

SRC_SCR field descriptions (continued)

Field	Description
1 sw_gpu_rst	Software reset for GPU **NOTE:** This is a self clearing bit. Once it is set to 1, the reset process will begin, and once it finishes, this bit will be self cleared. Software can determine that the reset has finished once this bit is cleared. Software can also configure SRC to generate interrupt once the software has finished. Please refer to SRC_SISR register for details. **NOTE:** The reset process will involve 8 GPU cycles before negating the GPU reset, to allow reset assertion to propagate into GPU. 0 do not assert GPU reset 1 assert GPU reset
0 warm_reset_ enable	WARM reset enable bit. WARM reset will be enabled only if warm_reset_enable bit is set. Otherwise all WARM reset sources will generate COLD reset. 0 WARM reset disabled 1 WARM reset enabled

60.7.2 SRC Boot Mode Register 1 (SRC_SBMR1)

The Boot Mode register (SBMR) contains bits that reflect the status of Boot Mode Pins of the chip. The reset value is configuration dependent (depending on boot/fuses/IO pads).

Address: 20D_8000h base + 4h offset = 20D_8004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BOOT_CFG4[7:0]								BOOT_CFG3[7:0]								BOOT_CFG2[7:0]								BOOT_CFG1[7:0]							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRC_SBMR1 field descriptions

Field	Description
31–24 BOOT_ CFG4[7:0]	Refer to fusemap.
23–16 BOOT_ CFG3[7:0]	Refer to fusemap.
15–8 BOOT_ CFG2[7:0]	Refer to fusemap.
BOOT_ CFG1[7:0]	Refer to fusemap.

60.7.3 SRC Reset Status Register (SRC_SRSR)

The SRSR is a write to one clear register which records the source of the reset events for the chip. The SRC reset status register will capture all the reset sources that have occurred. This register is reset on ipp_reset_b. This is a read-write register.

For bit[6-0] - writing zero does not have any effect. Writing one will clear the corresponding bit. The individual bits can be cleared by writing one to that bit. When the system comes out of reset, this register will have bits set corresponding to all the reset sources that occurred during system reset. Software has to take care to clear this register by writing one after every reset that occurs so that the register will contain the information of recently occurred reset.

Address: 20D_8000h base + 8h offset = 20D_8008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															warm_boot
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

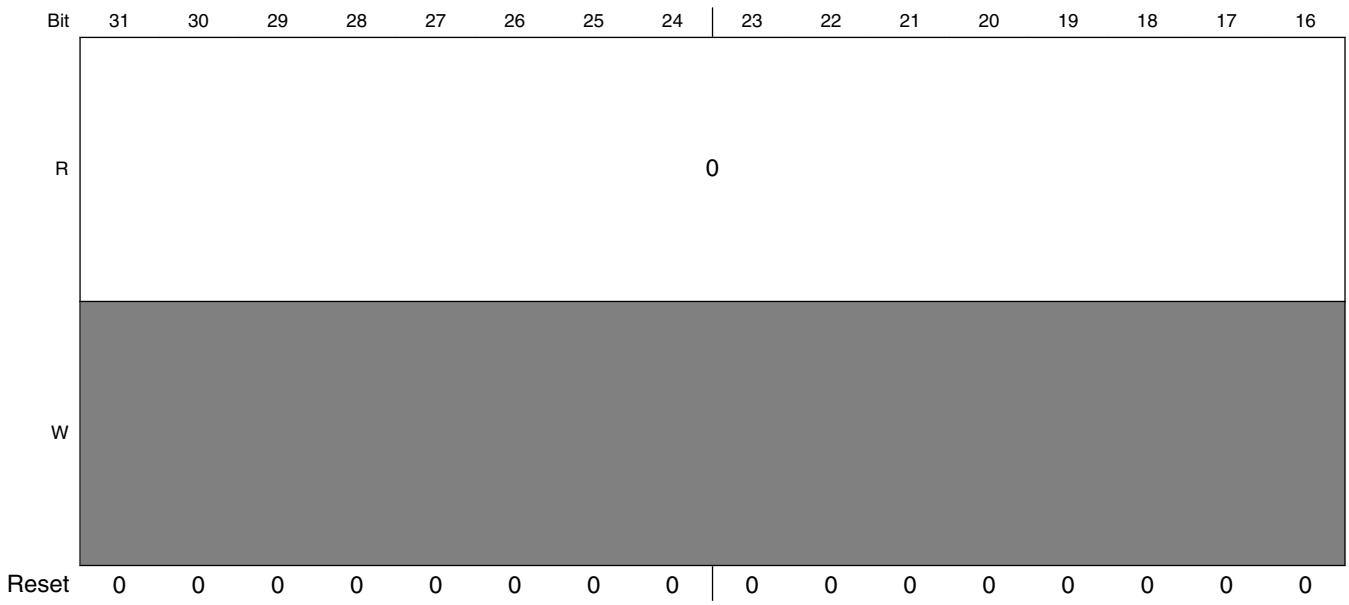
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0									jtag_sw_rst	jtag_rst_b	wdog_rst_b	ipp_user_reset_b	csu_reset_b	0	ipp_reset_b
W										w1c	w1c	w1c	w1c	w1c		w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

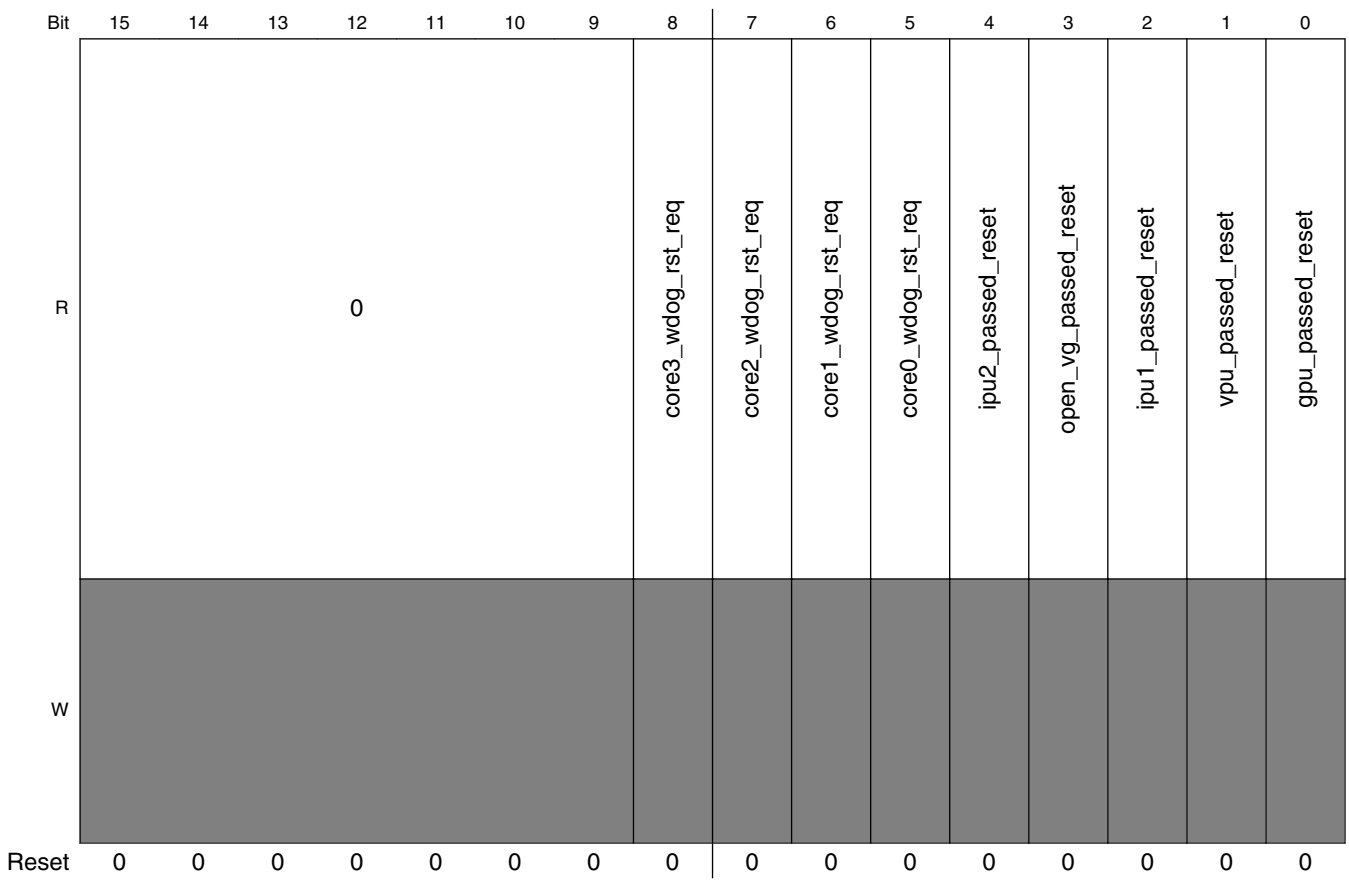
SRC_SRSR field descriptions

Field	Description
31–17 Reserved	This read-only field is reserved and always has the value 0.
16 warm_boot	WARM boot indication shows that WARM boot was initiated by software. This indicates to the software that it saved the needed information in the memory before initiating the WARM reset. In this case, software will set this bit to '1', before initiating the WARM reset. The warm_boot bit should be used as indication only after a warm_reset sequence. Software should clear this bit after warm_reset to indicate that the next warm_reset is not performed with warm_boot. Please refer to Reset Sequence and De-Assertion for details on warm_reset. 0 WARM boot process not initiated by software. 1 WARM boot initiated by software.
15–7 Reserved	This read-only field is reserved and always has the value 0.
6 jtag_sw_rst	JTAG software reset. Indicates whether the reset was the result of software reset from JTAG. 0 Reset is not a result of software reset from JTAG. 1 Reset is a result of software reset from JTAG.
5 jtag_rst_b	HIGH - Z JTAG reset. Indicates whether the reset was the result of HIGH-Z reset from JTAG. 0 Reset is not a result of HIGH-Z reset from JTAG. 1 Reset is a result of HIGH-Z reset from JTAG.
4 wdog_rst_b	IC Watchdog Time-out reset. Indicates whether the reset was the result of the watchdog time-out event. 0 Reset is not a result of the watchdog time-out event. 1 Reset is a result of the watchdog time-out event.
3 ipp_user_reset_b	Indicates whether the reset was the result of the ipp_user_reset_b qualified reset. 0 Reset is not a result of the ipp_user_reset_b qualified as COLD reset event. 1 Reset is a result of the ipp_user_reset_b qualified as COLD reset event.
2 csu_reset_b	Indicates whether the reset was the result of the csu_reset_b input. **NOTE:** If case the csu_reset_b occurred during a WARM reset process, during the phase that ipg_clk is not available yet, then the occurrence of CSU reset will not be reflected in this bit. 0 Reset is not a result of the csu_reset_b event. 1 Reset is a result of the csu_reset_b event.
1 Reserved	This read-only field is reserved and always has the value 0.
0 ipp_reset_b	Indicates whether reset was the result of ipp_reset_b pin (Power-up sequence) 0 Reset is not a result of ipp_reset_b pin. 1 Reset is a result of ipp_reset_b pin.

60.7.4 SRC Interrupt Status Register (SRC_SISR)

Address: 20D_8000h base + 14h offset = 20D_8014h



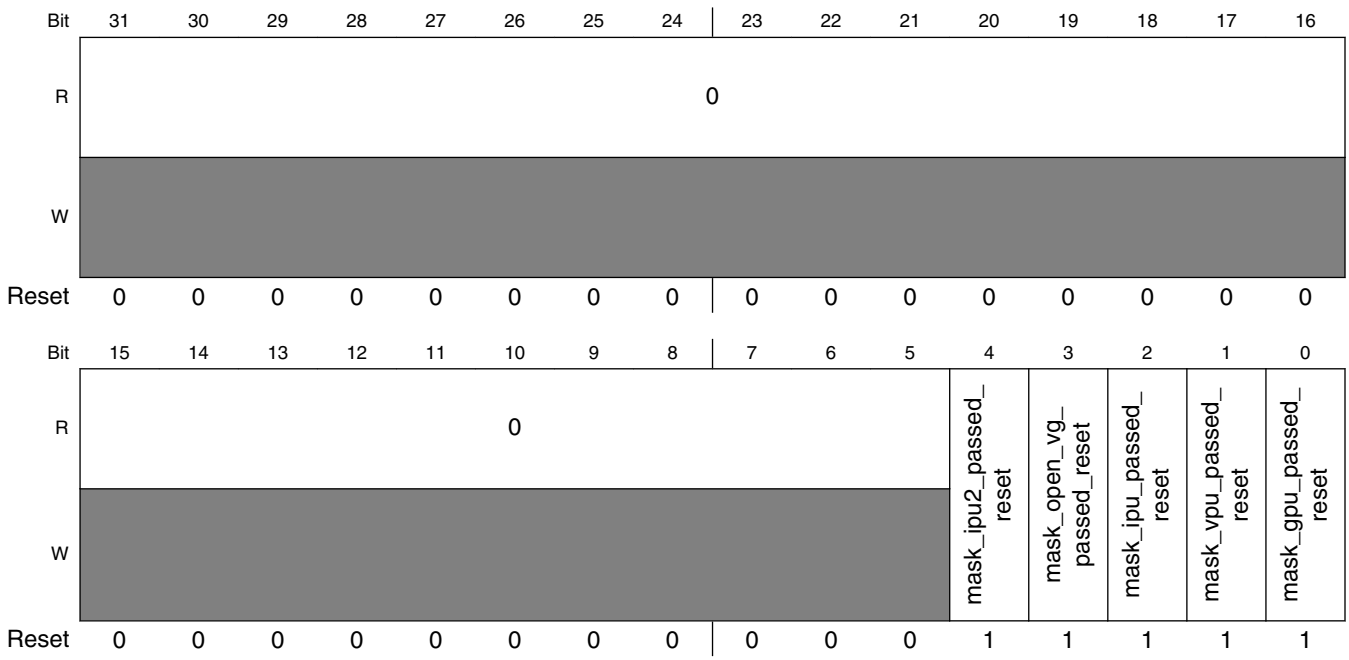


SRC_SISR field descriptions

Field	Description
31–9 Reserved	This read-only field is reserved and always has the value 0.
8 core3_wdog_rst_req	WDOG reset request from core3. Read-only status bit.
7 core2_wdog_rst_req	WDOG reset request from core2. Read-only status bit.
6 core1_wdog_rst_req	WDOG reset request from core1. Read-only status bit.
5 core0_wdog_rst_req	WDOG reset request from core0. Read-only status bit.
4 ipu2_passed_reset	Interrupt generated to indicate that ipu2 passed software reset and is ready to be used 0 interrupt generated not due to ipu2 passed reset 1 interrupt generated due to ipu2 passed reset
3 open_vg_passed_reset	Interrupt generated to indicate that open_vg passed software reset and is ready to be used 0 interrupt generated not due to open_vg passed reset 1 interrupt generated due to open_vg passed reset
2 ipu1_passed_reset	Interrupt generated to indicate that ipu passed software reset and is ready to be used 0 interrupt generated not due to ipu passed reset 1 interrupt generated due to ipu passed reset
1 vpu_passed_reset	Interrupt generated to indicate that VPU passed software reset and is ready to be used 0 interrupt generated not due to VPU passed reset 1 interrupt generated due to VPU passed reset
0 gpu_passed_reset	Interrupt generated to indicate that GPU passed software reset and is ready to be used 0 interrupt generated not due to GPU passed reset 1 interrupt generated due to GPU passed reset

60.7.5 SRC Interrupt Mask Register (SRC_SIMR)

Address: 20D_8000h base + 18h offset = 20D_8018h



SRC_SIMR field descriptions

Field	Description
31–5 Reserved	This read-only field is reserved and always has the value 0.
4 mask_ipu2_passed_reset	mask interrupt generation due to ipu2 passing reset 0 do not mask interrupt due to ipu2 passed reset - interrupt will be created 1 mask interrupt due to ipu2 passed reset
3 mask_open_vg_passed_reset	mask interrupt generation due to open_vg passed reset 0 do not mask interrupt due to open_vg passed reset - interrupt will be created 1 mask interrupt due to open_vg passed reset
2 mask_ipu_passed_reset	mask interrupt generation due to ipu passed reset 0 do not mask interrupt due to ipu passed reset - interrupt will be created 1 mask interrupt due to ipu passed reset
1 mask_vpu_passed_reset	mask interrupt generation due to VPU passed reset 0 do not mask interrupt due to VPU passed reset - interrupt will be created 1 mask interrupt due to VPU passed reset
0 mask_gpu_passed_reset	mask interrupt generation due to GPU passed reset 0 do not mask interrupt due to GPU passed reset - interrupt will be created 1 mask interrupt due to GPU passed reset

60.7.6 SRC Boot Mode Register 2 (SRC_SBMR2)

The Boot Mode register (SBMR), contains bits that reflect the status of Boot Mode Pins of the chip. The default values for those bits depends on the values of pins/fuses during reset sequence, hence the question mark on their default value.

Address: 20D_8000h base + 1Ch offset = 20D_801Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0						BMOD[1:0]		0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0											BT_FUSE_SEL	DIR_BT_DIS	0	SEC_CONFIG[1:0]	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRC_SBMR2 field descriptions

Field	Description
31–26 Reserved	This read-only field is reserved and always has the value 0.
25–24 BMOD[1:0]	BMOD[1:0] shows the latched state of the BOOT_MODE1 and BOOT_MODE0 signals on the rising edge of POR_B. See the Boot mode pin settings section of System Boot.
23–5 Reserved	This read-only field is reserved and always has the value 0.
4 BT_FUSE_SEL	BT_FUSE_SEL (connected to gpio bt_fuse_sel) shows the state of the BT_FUSE_SEL fuse. See Fusemap for additional information on this fuse.
3 DIR_BT_DIS	DIR_BT_DIS shows the state of the DIR_BT_DIS fuse. See Chapter 5, Fusemap for additional information on this fuse.
2 Reserved	This read-only field is reserved and always has the value 0.
SEC_ CONFIG[1:0]	SECONFIG[1] shows the state of the SECONFIG[1] fuse. See Fusemap for additional information on this fuse. SECONFIG[0] shows the state of the SECONFIG[0] fuse. This fuse is shown as reserved in Fusemap (address 0x440[1]) because it does not have a user-relevant function.

60.7.7 SRC General Purpose Register 1 (SRC_GPR1)

Address: 20D_8000h base + 20h offset = 20D_8020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PERSISTENT_ENTRY0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRC_GPR1 field descriptions

Field	Description
PERSISTENT_ ENTRY0	Holds entry function for core0 for waking-up from low power mode. The SRC ensures that the register value will persist across system resets.

60.7.8 SRC General Purpose Register 2 (SRC_GPR2)

Address: 20D_8000h base + 24h offset = 20D_8024h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PERSISTENT_ARG0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRC_GPR2 field descriptions

Field	Description
PERSISTENT_ ARG0	Holds argument of entry function for core0 for waking-up from low power mode. The SRC ensures that the register value will persist across system resets.

60.7.9 SRC General Purpose Register 3 (SRC_GPR3)

Address: 20D_8000h base + 28h offset = 20D_8028h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PERSISTENT_ENTRY1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRC_GPR3 field descriptions

Field	Description
PERSISTENT_ENTRY1	Holds entry function for core1. The SRC ensures that the register value will persist across system resets.

60.7.10 SRC General Purpose Register 4 (SRC_GPR4)

Address: 20D_8000h base + 2Ch offset = 20D_802Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PERSISTENT_ARG1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRC_GPR4 field descriptions

Field	Description
PERSISTENT_ARG1	Holds argument of entry function for core1. The SRC ensures that the register value will persist across system resets.

60.7.11 SRC General Purpose Register 5 (SRC_GPR5)

Address: 20D_8000h base + 30h offset = 20D_8030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PERSISTENT_ENTRY2																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRC_GPR5 field descriptions

Field	Description
PERSISTENT_ENTRY2	Holds entry function for core2 (i.MX 6Quad only). The SRC ensures that the register value will persist across system resets.

60.7.12 SRC General Purpose Register 6 (SRC_GPR6)

Address: 20D_8000h base + 34h offset = 20D_8034h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PERSISTENT_ARG2																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRC_GPR6 field descriptions

Field	Description
PERSISTENT_ARG2	Holds argument of entry function for core2 (i.MX 6Quad only). The SRC ensures that the register value will persist across system resets.

60.7.13 SRC General Purpose Register 7 (SRC_GPR7)

Address: 20D_8000h base + 38h offset = 20D_8038h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PERSISTENT_ENTRY3																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRC_GPR7 field descriptions

Field	Description
PERSISTENT_ENTRY3	Holds entry function for core3 (i.MX 6Quad only). The SRC ensures that the register value will persist across system resets.

60.7.14 SRC General Purpose Register 8 (SRC_GPR8)

Address: 20D_8000h base + 3Ch offset = 20D_803Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PERSISTENT_ARG3																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

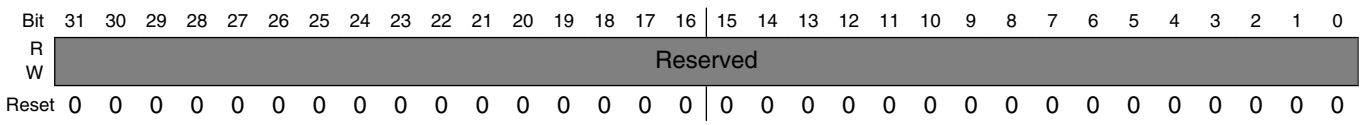
SRC_GPR8 field descriptions

Field	Description
PERSISTENT_ARG3	Holds argument of entry function for core3 (i.MX 6Quad only). The SRC ensures that the register value will persist across system resets.

60.7.15 SRC General Purpose Register 9 (SRC_GPR9)

Reserved for Internal Use.

Address: 20D_8000h base + 40h offset = 20D_8040h

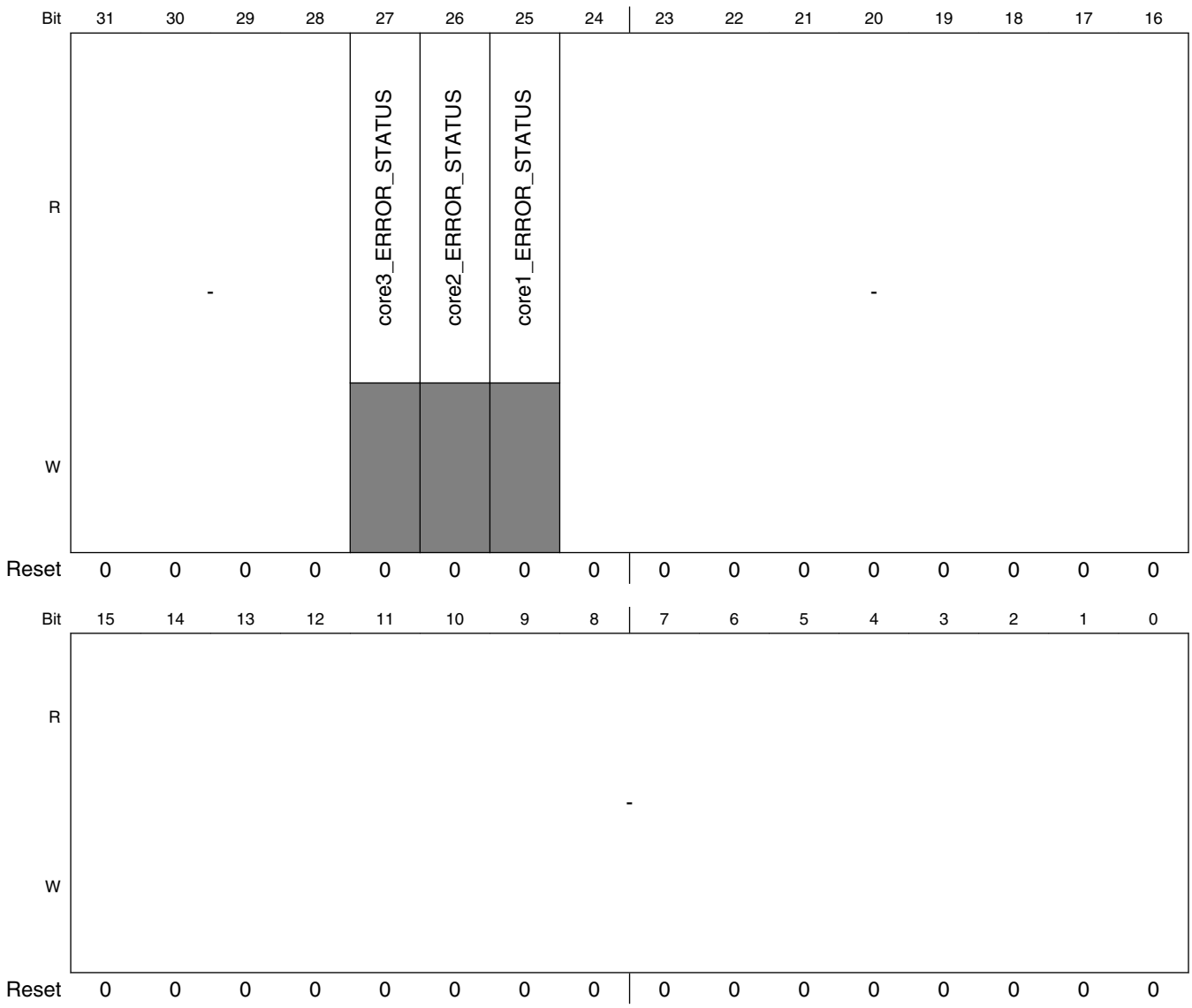


SRC_GPR9 field descriptions

Field	Description
-	This field is reserved. Reserved.

60.7.16 SRC General Purpose Register 10 (SRC_GPR10)

Address: 20D_8000h base + 44h offset = 20D_8044h



SRC_GPR10 field descriptions

Field	Description
31–28 -	Read/write bit, for general purpose **NOTE:** Reset only by POR
27 core3_ERROR_ STATUS	core3 error status bit (i.MX 6Quad Only).

Table continues on the next page...

SRC_GPR10 field descriptions (continued)

Field	Description
26 core2_ERROR_ STATUS	core2 error status bit (i.MX 6Quad Only).
25 core1_ERROR_ STATUS	core1 error status bit.
-	Read/write bits, for general purpose **NOTE:** Reset only by POR

Chapter 61
Synchronous Serial Interface (SSI)

61.1 Overview

This block guide presents the Synchronous Serial Interface (SSI), and discusses the architecture, the programming model, the operating modes, and initialization of SSI.

The SSI is a full-duplex, serial port that allows the chip to communicate with a variety of serial devices. These serial devices can be standard CODer-DECoder (CODECs), Digital Signal Processors (DSPs), microprocessors, peripherals, and popular industry audio codecs that implement the inter-IC sound bus standard (I2S) and Intel AC97 standard.

SSI is typically used to transfer samples in a periodic manner. The SSI consists of independent transmitter and receiver sections with independent clock generation and frame synchronization.

The Figure 61-1 illustrates the organization of the SSI. It consists of control registers to set up the port, status register, separate transmit and receive circuits with FIFO registers, and separate serial clock and frame sync generation for the transmit and receive sections. The second set of Tx and Rx FIFOs, replicates the logic used for the first set of FIFOs.

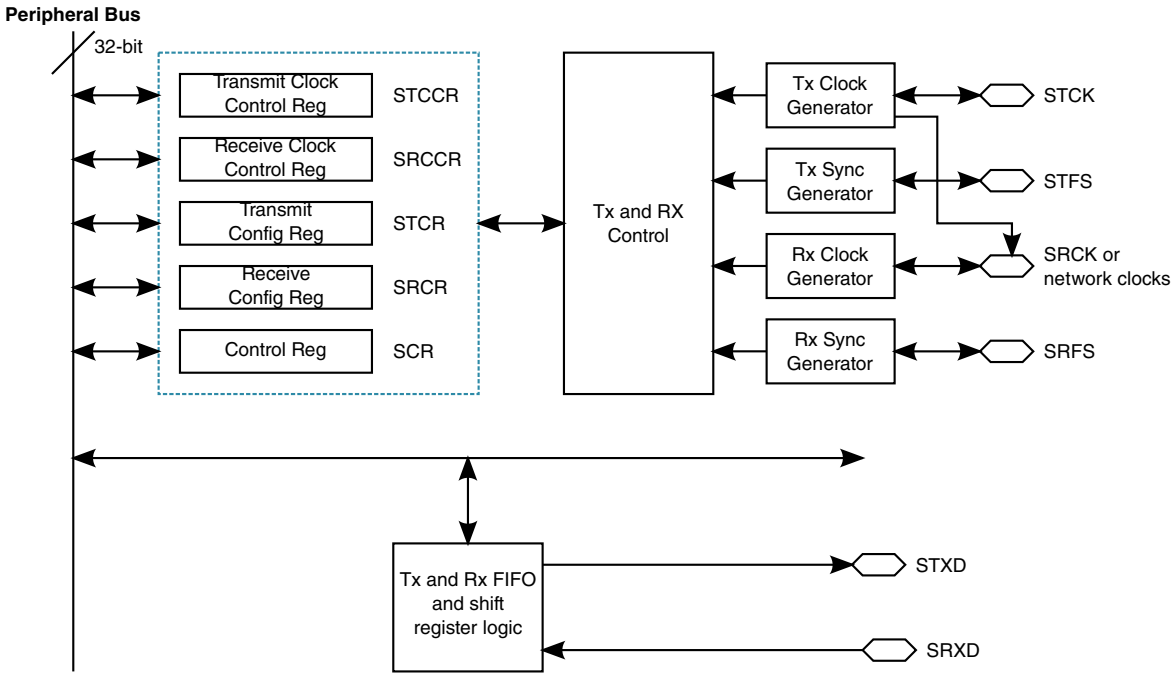


Figure 61-1. SSI Block Diagram

61.1.1 Features

The SSI includes the following features:

- Independent (asynchronous) or shared (synchronous) transmit and receive sections with separate or shared internal/external clocks and frame syncs, operating in Master or Slave mode.
- Normal mode operation using frame sync
- Network mode operation allowing multiple devices to share the port with as many as thirty-two time slots
- Gated Clock mode operation requiring no frame sync
- 2 sets of Transmit and Receive FIFOs. Each of the four FIFOs is 15x32 bits. The two sets of Tx/Rx FIFOs can be used in Network mode to provide 2 independent channels for transmission and reception (this mode is named as two-channel mode in the following descriptions)
- Programmable data interface modes such as I2S, LSB, MSB aligned
- Programmable word length (8, 10, 12, 16, 18, 20, 22 or 24 bits)
- Program options for frame sync and clock generation
- Programmable I2S modes (Master, Slave or Normal). Maximum audio sampling rate is 196kHz. (Note that maximum sampling rate depends on IPG frequency.) Minimum audio sampling rate is 8kHz. Network clock (as an oversampling clock to external device) available as output from SRCK in I2S Master mode

- AC97 support. Max frame rate is 48kHz. Min frame rate is 8kHz.
- Completely separate clock and frame sync selections for the receive and transmit sections. In AC97 standard, the clock is taken from an external source and frame sync is generated internally.
- SSI's system clock (generated inside CCM) can be used in I2S Transmitter Master mode. This system clock is also available as source clock for output SRCK in master mode, when operated in sync mode.
- Programmable internal clock divider
- Time Slot Mask Registers for reduced ARM platform overhead (for both Tx and Rx)
- SSI power-down feature

61.1.2 Modes of Operation

SSI has the following basic operating modes.

- Normal Mode : Asynchronous protocol, Synchronous protocol
  - Normal Mode Transmit
  - Normal Mode Receive
- Network Mode : Asynchronous protocol, Synchronous protocol
  - Network Mode Transmit
  - Network Mode Receive
- Gated Clock Mode : Synchronous protocol only
- I2S Mode
- AC97 Mode
  - AC97 Fixed Mode (SSI.SACNT[1]=0)
  - AC97 Variable Mode (SSI.SACNT[1]=1)

61.2 External Signal Description

61.2.1 Signals Overview

The Synchronous Serial Interface (SSI) can be connected directly to the external pins or through the Digital Audio Multiplexer (AUDMUX). Refer to the AUDMUX chapter for programming details of the various multiplexing options.

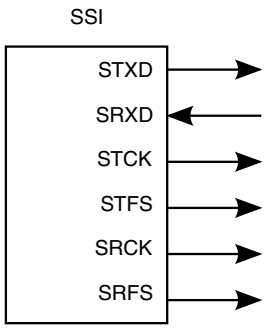
Table 61-1. Off-Chip Block Signals

Name	I/O	Function	Reset State	Pull up
SRCK	I/O	Serial Receive Clock. SRCK can be used as either an input or an output. This clock signal is used by the receiver in asynchronous mode and is always continuous. During synchronous mode, the STCK port is used instead for clocking in data. In SSI synchronous modes, this port can be used as an output for the network clock (oversampling clock) . In I2S master mode, this signal can be used to output the network clock to an external CODEC.	0	Passive
SRFS	I/O	Serial Receive Frame Sync. The SRFS port can be used as either an input or an output. The frame sync is used by the receiver to synchronize the transfer of data. The frame sync signal can be one bit or one word in length and can occur one bit before the transfer of data or right at the transfer of data. If SRFS is configured as an input, the external device should drive SRFS during the rising edge of STCK or SRCK.	0	Passive
SRXD	I	Serial Receive Data. The SRXD port is an input and is used to bring serial data into the Receive Data Shift Register.	-	-
STCK	I/O	Serial Transmit Clock. The STCK port can be used as either an input or an output. This clock signal is used by the transmitter and can be either continuous or gated. During Gated Clock mode, data on the STCK port is valid only during the transmission of data, otherwise it is pulled to the inactive state. In Synchronous mode, this port is used by both the transmit and receive sections.	0	Passive
STFS	I/O	Serial Transmit Frame Sync. The STFS port can be used as either an input or an output. The frame sync is used by the transmitter to synchronize the transfer of data. The frame sync signal can be one bit or one word in length and can occur one bit before the transfer of data or right at the transfer of data. In Synchronous mode, this port is used by both the transmit and receive sections. In Gated Clock mode, frame sync signals are not used. If STFS is configured as an input, the external device should drive STFS during the rising edge of STCK if TSCKP is positive edge triggered. The external device should drive STFS during the falling edge of STCK if TSCKP is negative edge triggered.	0	Passive
STXD	O	Serial Transmit Data. The STXD port is an output and transmits data from the Serial Transmit Shift Register. The STXD port is an output port when data is being transmitted and is disabled between data word transmissions and on the trailing edge of the bit clock after the last bit of a word is transmitted.	0	Passive

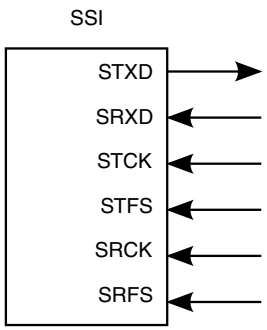
The following figure shows the main SSI configurations. These ports support all transmit and receive functions with continuous or gated clock as shown.

NOTE

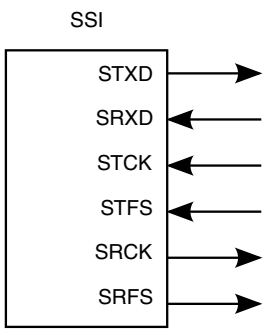
Gated clock implementations do not require the use of the frame sync ports (STFS and SRFS).



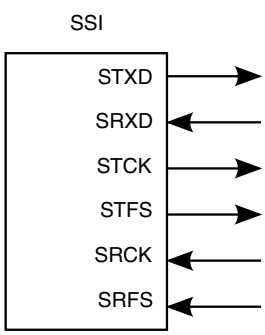
SSI Internal Continuous Clock for TX/RX (RXDIR = 1, TXDIR = 1, RFDIR = 1, TFDIR = 1, SYN = 0)



SSI External Continuous Clock for TX/RX (RXDIR = 0, TXDIR = 0, RFDIR = 0, TFDIR = 0, SYN = 0)



SSI Internal Continuous Clock for RX (RXDIR = 1, TXDIR = 0, RFDIR = 1, TFDIR = 0, SYN = 0)
SSI External Continuous Clock for TX



SSI Internal Continuous Clock for TX (RXDIR = 0, TXDIR = 1, RFDIR = 0, TFDIR = 1, SYN = 0)
SSI External Continuous Clock for RX

Figure 61-2. Asynchronous (SYN=0) SSI Configurations-Continuous Clock

See the following figure for an example of the port signals for an 8-bit data transfer. Continuous and gated clock signals are shown, as well as the bit-length frame sync signal and the word-length frame sync signal.

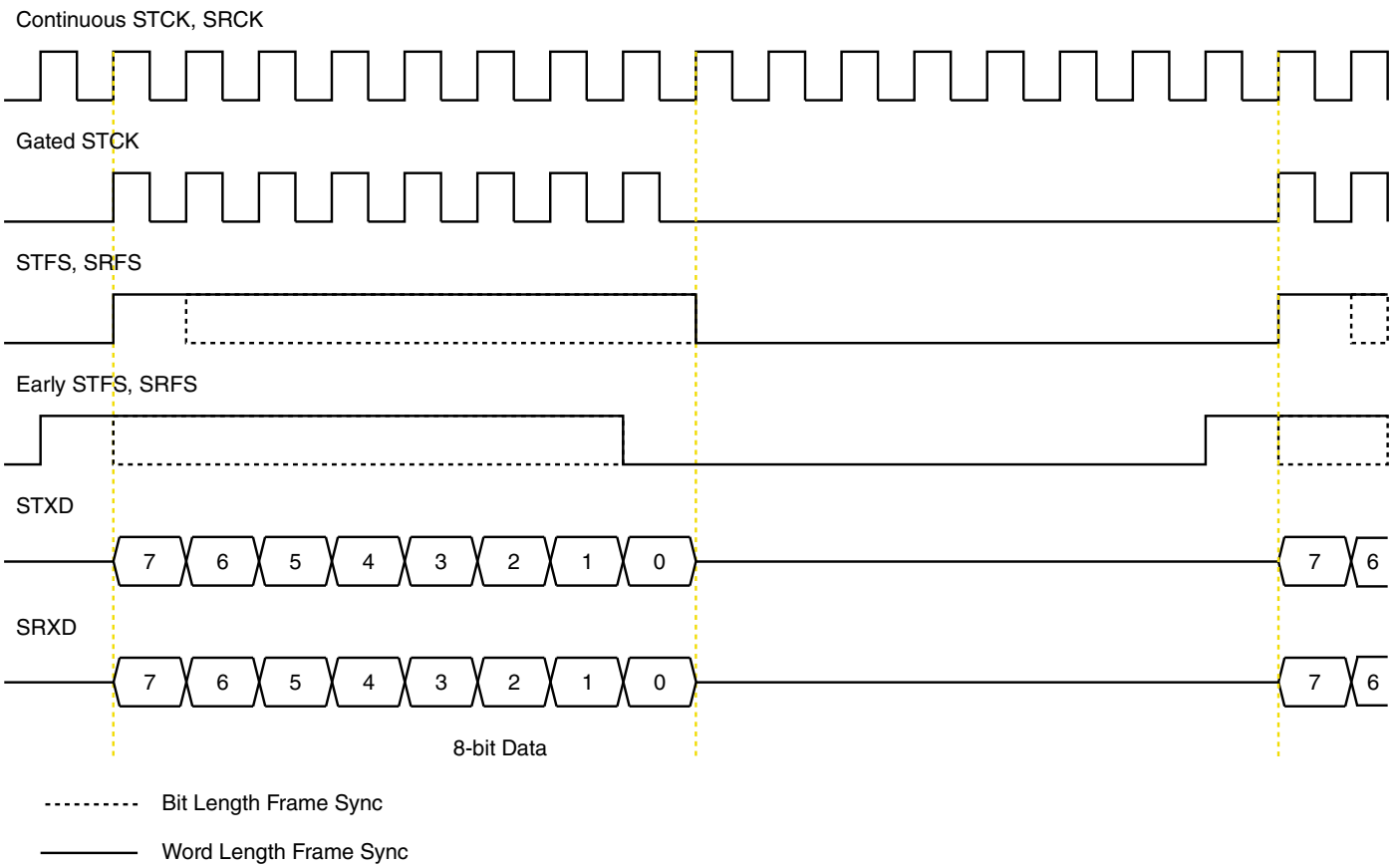


Figure 61-3. Serial Clock and Frame Sync Timing

See the table below for list of clock pin configurations.

Table 61-2. Clock Pin Configurations

SYN	RXDIR	TXDIR	RFDIR	TFDIR	SRCK	STCK	SRFS	STFS
Asynchronous Mode								
0	0	0	0	0	RCK in	TCK in	RFS in	TFS in
0	0	0	0	1	RCK in	TCK in	RFS in	TFS out
0	0	0	1	0	RCK in	TCK in	RFS out	TFS in
0	0	0	1	1	RCK in	TCK in	RFS out	TFS out
0	0	1	0	0	RCK in	TCK out	RFS in	TFS in
0	0	1	0	1	RCK in	TCK out	RFS in	TFS out
0	0	1	1	0	RCK in	TCK out	RFS out	TFS in
0	0	1	1	1	RCK in	TCK out	RFS out	TFS out
0	1	0	0	0	RCK out	TCK in	RFS in	TFS in

Table continues on the next page...

Table 61-2. Clock Pin Configurations (continued)

SYN	RXDIR	TXDIR	RFDIR	TFDIR	SRCK	STCK	SRFS	STFS
0	1	0	0	1	RCK out	TCK in	RFS in	TFS out
0	1	0	1	0	RCK out	TCK in	RFS out	TFS in
0	1	0	1	1	RCK out	TCK in	RFS out	TFS out
0	1	1	0	0	RCK out	TCK out	RFS in	TFS in
0	1	1	0	1	RCK out	TCK out	RFS in	TFS out
0	1	1	1	0	RCK out	TCK out	RFS out	TFS in
0	1	1	1	1	RCK out	TCK out	RFS out	TFS out
Synchronous Mode								
1	0	0	x	0	-	CK in	-	FS in
1	0	0	x	1	-	CK in	-	FS out
1	0	1	x	0	-	CK out	-	FS in
1	0	1	x	1	-	CK out	-	FS out
1	1	0	x	x	-	Gated in	-	-
1	1	1	x	x	-	Gated out	-	-

61.3 Clocks

The table found here describes the clock sources for SSI.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 61-3. SSI Clocks

Clock name	Clock Root	Description
ccm_ssi_clk	ssi_clk_root	Module / system clock for bit clock generation
ipg_clk	ipg_clk_root	Peripheral clock
ipg_clk_s	ipg_clk_root	Peripheral access clock

61.4 SSI Transmit FIFO 0 & 1 Registers

The SSI Transmit FIFO registers are 15x32-bit registers. These registers are not directly accessible by the end user. Transmit Shift Register (TXSR) receives its values from these FIFO registers. Transmitted data is first-in-first-out.

When the Transmit Interrupt Enable (TIE) bit in the SIER and either of the Transmit FIFO Empty Enable (TFE0 or 1) bits in the SIER are set, an interrupt is asserted whenever the number of empty slots exceed or are equal to the selected threshold value of corresponding Tx-FIFO. The threshold value is contained in the corresponding Transmit FIFO Watermark (TFWM0 or 1) field in the SSI FIFO Control/Status Register (SFCSR).

61.5 SSI Transmit Shift Register (TXSR)

The SSI Transmit Shift Register (TXSR) is a 24-bit shift register that contains the data being transmitted.

This register is not directly accessible by the end user. When a continuous clock is used, data is shifted out to the Serial Transmit Data (STXD) port by the selected (internal/external) bit clock when the associated (internal/external) frame sync is asserted. When a gated clock is used, data is shifted out to the STXD port by the selected (internal/external) gated clock. The Word Length control bits (WL[3:0]) in the SSI Transmit and Receive Clock Control Register (STCCR) determine the number of bits to be shifted out of the TXSR before it is considered empty and can be written to again. This word length can be 8, 10, 12, 16, 18, 20, 22 or 24 bits. The data to be transmitted occupies the most significant portion of the shift register if TXBIT0 is '0', otherwise it occupies the least significant portion. The unused portion of the register is ignored. Data is always shifted out of this register with the Most Significant Bit (MSB) first when the SHFD bit of the STCR is cleared. If this bit is set, the Least Significant Bit (LSB) is shifted out first. The figures below show the transmitter loading and shifting operation. The figures show the working for some WL values, the same can be extended for other values.

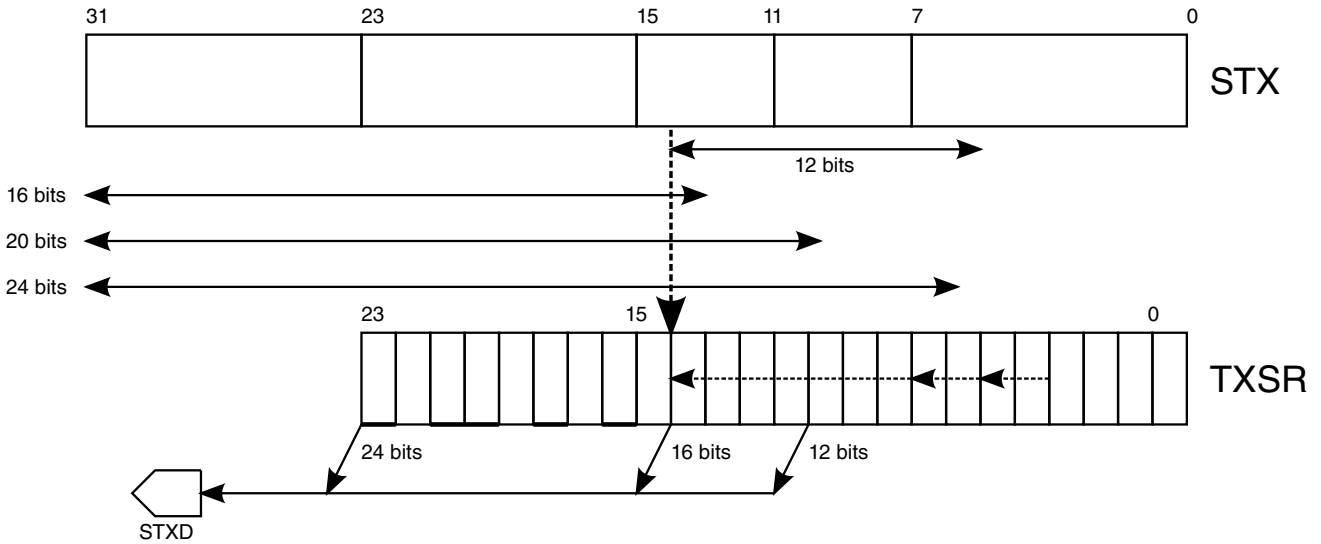


Figure 61-4. Transmit Data Path (TXBIT0=0, TSHFD=0) (MSB Alignment)

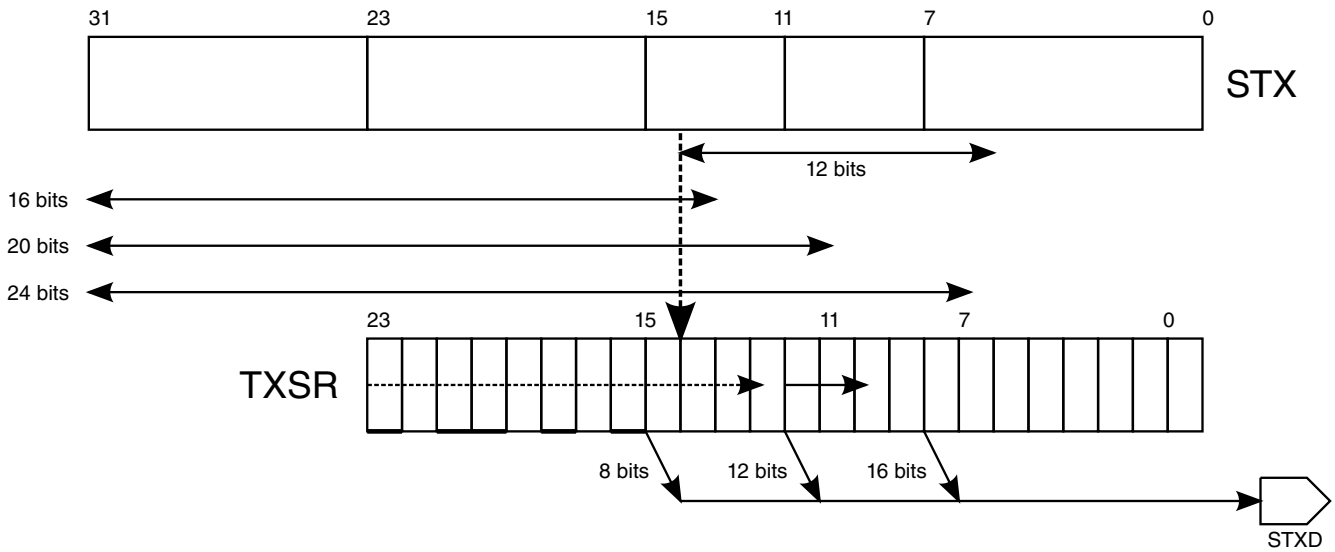


Figure 61-5. Transmit Data Path (TXBIT0=0, TSHFD=1) (MSB Alignment)

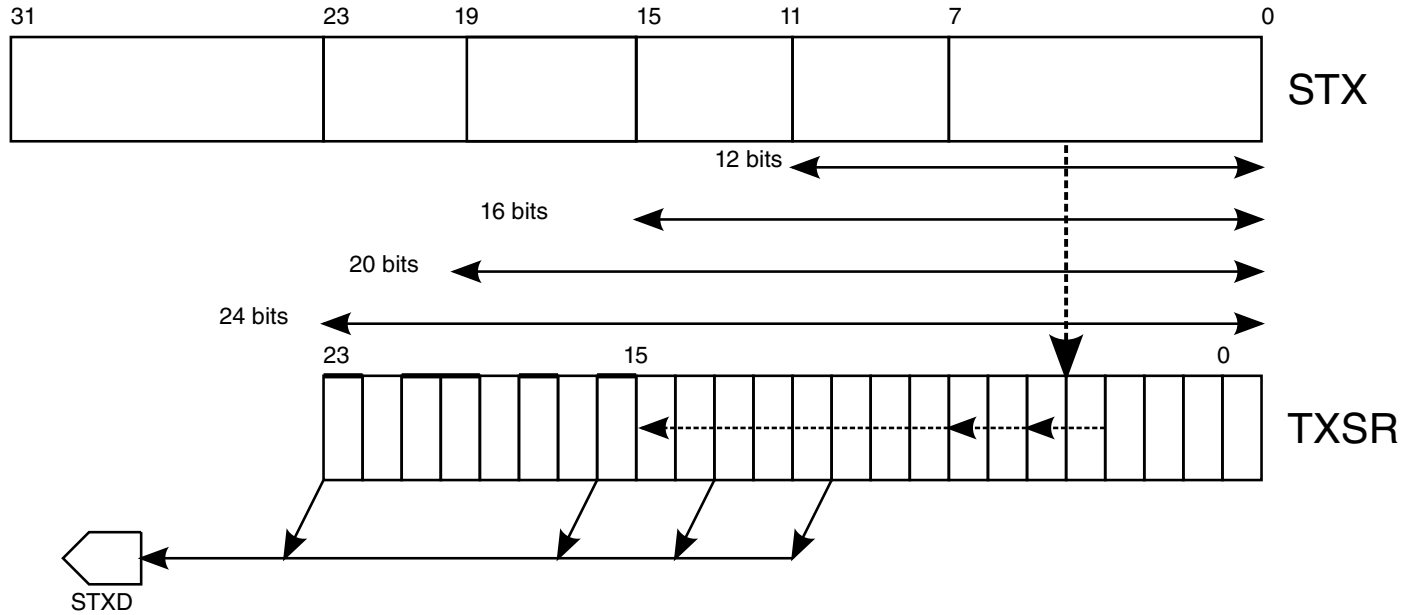


Figure 61-6. Transmit Data Path (TXBIT0=1, TSHFD=0) (LSB Alignment)

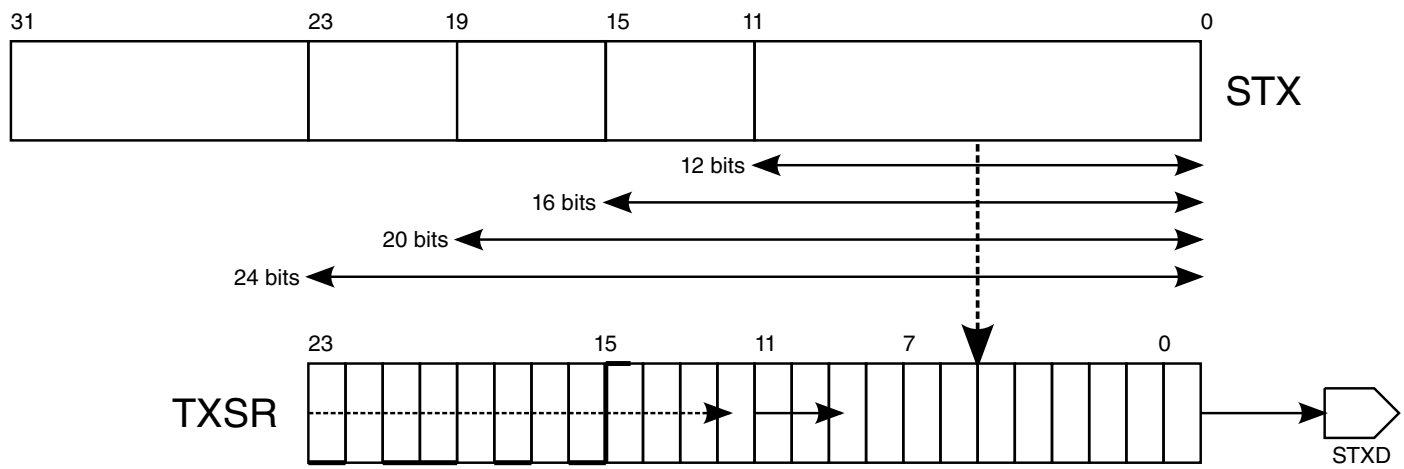


Figure 61-7. Transmit Data Path (TXBIT0=1, TSHFD=1) (LSB Alignment)

61.6 SSI Receive FIFO 0 and 1 Registers

The SSI Receive FIFO registers are 15x32-bit registers. These registers are not directly accessible by the end user. These FIFO registers receive data from the Receive Shift Register (RXSR). Received data is first-in-first-out.

When the Receive Interrupt Enable (RIE) bit in the SIER and either of the Receive FIFO Full Enable (RFF0_EN or RFF1_EN) bits in the SIER are set, an interrupt is asserted whenever the number of full slots exceeds or is equal to the selected threshold value of corresponding Rx-FIFO. The threshold value is contained in the corresponding Receive FIFO Watermark (RFWM0 or 1) field in the SSI FIFO Control/Status Register (SFCSR).

61.7 SSI Receive Shift Register (RXSR)

The SSI Receive Shift Register (RXSR) is a 24-bit, shift register that receives incoming data from the serial receive data SRXD port.

This register is not directly accessible by the end user. When a continuous clock is used, data is shifted in by the selected (internal/external) bit clock when the associated (internal/external) frame sync is asserted. When a gated clock is used, data is shifted in by the selected (internal/external) gated clock. Data is assumed to be received MSB first if the SHFD bit of the SSI.SRCR is cleared. If this bit is set, the data is received LSB first. Data is transferred to the appropriate SSI Receive Data Register (SRX0/1) or Receive FIFOs (if the receive FIFO is enabled and the corresponding SRX is full) after 8, 10, 12, 16, 18, 20, 22 or 24 bits have been shifted in depending on the WL[3:0] control bits. For receiving less than 24 bits of data, LSB bits are appended with zero. The figures below show the receiver loading and shifting operation. These figures show the operation for several values of WL and the same can be extended for other values.

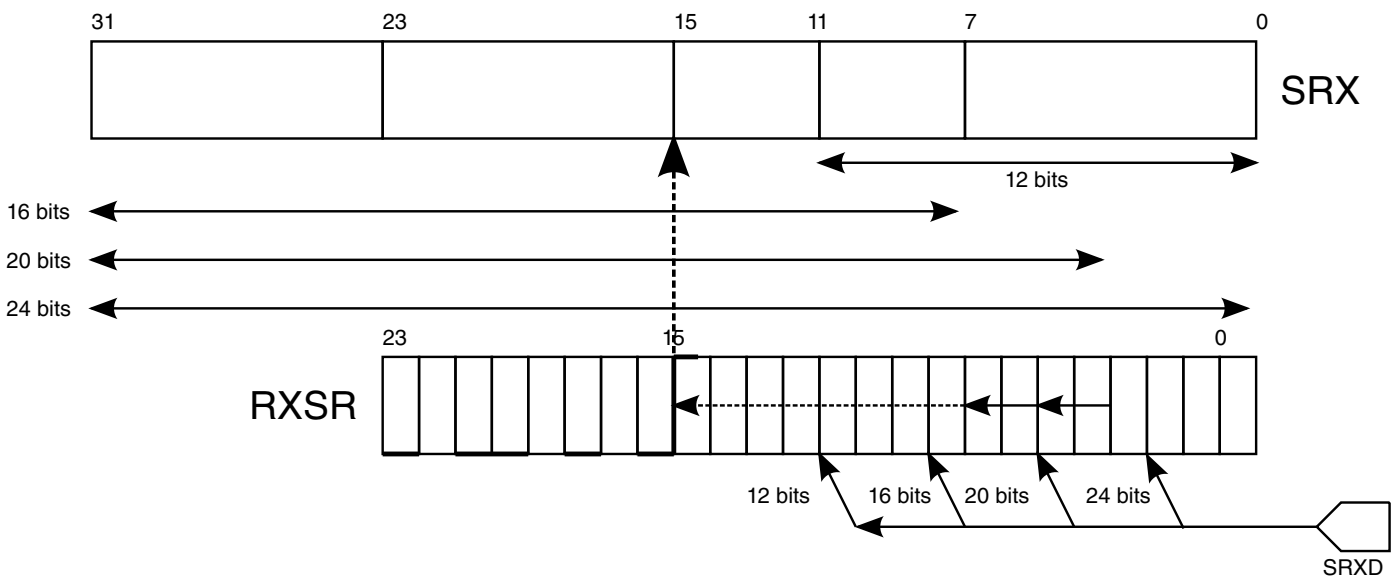


Figure 61-8. Receive Data Path (RXBIT0=0, RSHFD=0) (MSB Alignment)

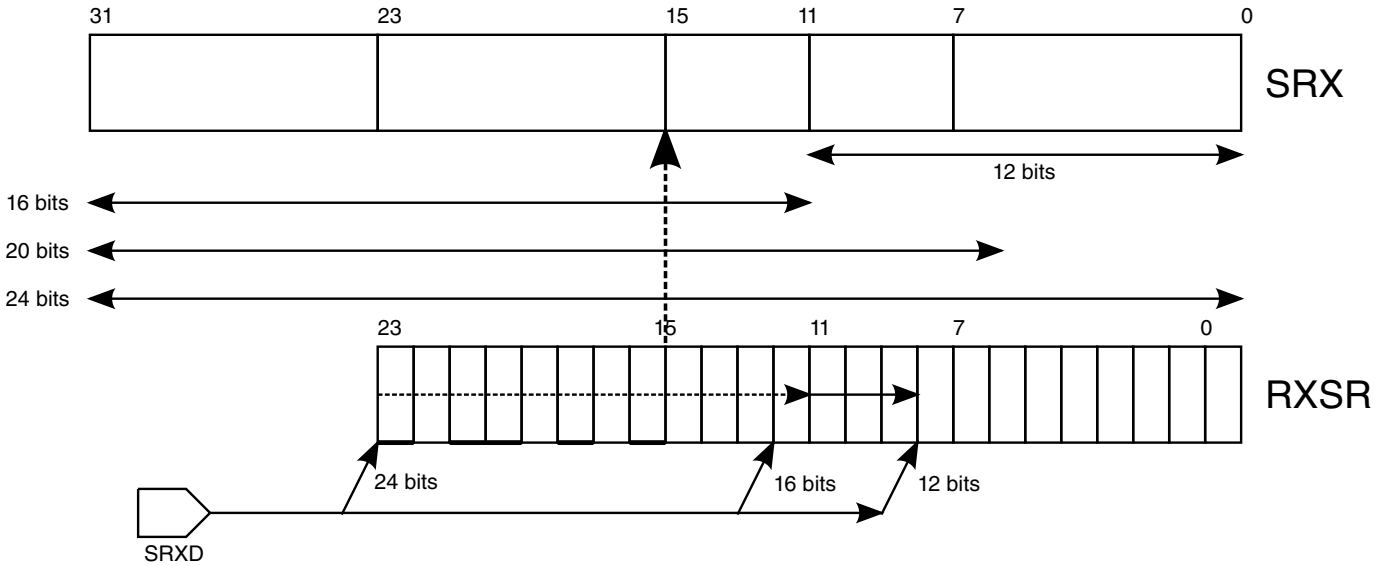


Figure 61-9. Receive Data Path (RXBIT0=0, RSHFD=1) (MSB Alignment)

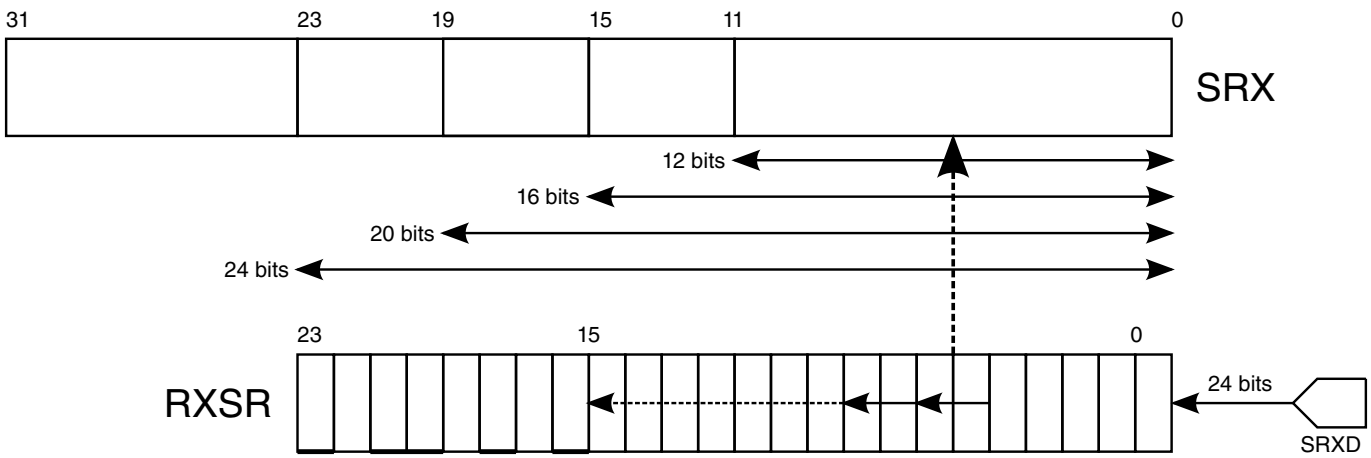


Figure 61-10. Receive Data Path (RXBIT0=1, RSHFD=0) (LSB Alignment)

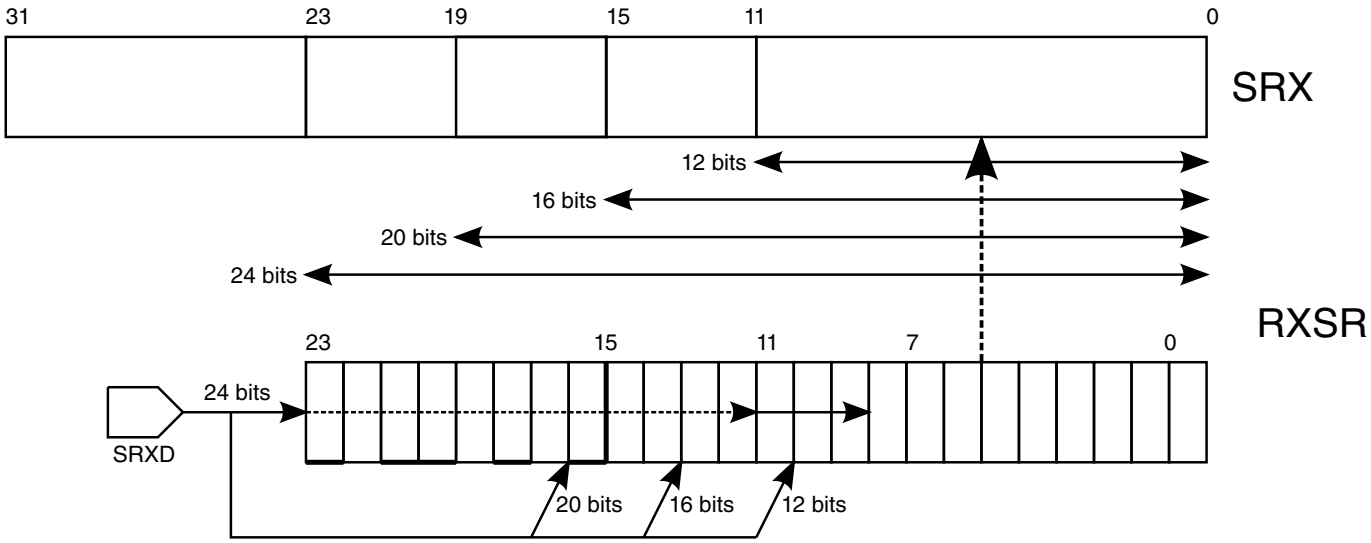


Figure 61-11. Receive Data Path (RXBIT0=1, RSHFD=1) (LSB Alignment)

61.8 Functional Description

This section provides a complete functional description of the block.

61.8.1 Operating Modes

Different modes can be programmed by several bits in the SSI control registers.

See the table below for the list of SSI operating modes and some of the typical applications in which they can be used:

Table 61-4. SSI Operating Modes

TX, RX Sections	Serial Clock	Mode	Typical Application
Asynchronous	Continuous	Normal	Multiple synchronous CODECs
Asynchronous	Continuous	Network	TDM CODEC or DSP networks
Synchronous	Continuous	Normal	Multiple synchronous CODECs
Synchronous	Continuous	Network	TDM CODEC or DSP network
Synchronous	Gated	Normal	SPI-type devices; DSP to ARM platform

The transmit and receive sections of the SSI can be synchronous or asynchronous. In Synchronous mode, the transmitter and the receiver use a common clock and frame synchronization signal. Masking of slots for Transmit and Receive section can differ in

synchronous mode. Also the shifting of data in independent and for receive section depends on RXBIT0 and RSHFD bits in SSI.SRCR register. In Asynchronous mode, the transmitter and receiver each has its own clock and frame synchronization signals.

Normal or Network mode can be selected. In Normal mode, the SSI functions with one data word of I/O per frame. In Network mode, any number from two to thirty-two data words of I/O per frame can be used. Network mode is typically used in time division multiplex networks with other processors or CODECs, allowing interface to time division multiplexed networks without additional logic. Gated clock mode option can be selected in Normal synchronous mode only. During Gated clock mode the clock is not-continuous and runs only during data-transmission. These distinctions result in the basic operating modes that allow the SSI to communicate with a wide variety of devices.

The SSI supports both Normal and Network modes, and these can be selected independently of whether the transmitter and receiver are synchronous or asynchronous. Typically these protocols are used in a periodic manner, where data is transferred at regular intervals, such as at the sampling rate of an external CODEC. Both modes use the concept of a frame. The beginning of the frame is marked with a frame sync when programmed with continuous clock. The frame sync occurs at a periodic interval. The length of the frame is determined by the DC[4:0] bits in either the SSI.SRCCR or SSI.STCCR register, depending on whether data is being transmitted or received. The number of words transferred per frame depends on the mode of the SSI.

In Normal mode, one data word is transferred per frame. In Network mode, the frame is divided into anywhere between two and thirty-two time slots, where in each time slot one data word can optionally be transferred.

Apart from the above basic modes of operation, SSI supports the following modes which require some specific programming.

- I2S mode
- AC97 mode
    - AC97 Fixed mode
    - AC97 Variable mode

In (non-I2S) slave modes (external frame sync), the programmed word length setting of the SSI should be equal to the word length setting of the master. In I2S slave mode, the programmed word length setting of the SSI can be lesser than or equal to the word length setting of the I2S master (external CODEC).

In slave modes, the programmed frame length setting (DC bits) of the SSI can be lesser than or equal to the frame length setting of the master (external CODEC).

The following sections provide detailed descriptions of the above modes.

61.8.1.1 Normal Mode

Normal mode is the simplest mode of the SSI. It is used to transfer data in one time slot per frame.

A time slot is a unit of data and the WL[3:0] bits define the number of bits in a time slot. In Continuous Clock mode, a frame sync occurs at the beginning of each frame. The length of the frame is determined by the following factors:

- The period of the Serial Bit Clock (DIV2, PSR, PM[7:0] bits for internal clock or the frequency of the external clock on the STCK port)
- The number of bits per time slot (WL[3:0] bits)
- The number of time slots per frame (DC[4:0] bits)

If Normal mode is configured with more than one time slot per frame, data is transferred only in the first time slot. No data is transferred in subsequent time slots. In Normal mode, DC[4:0] values corresponding to more than a single time slot in a frame, only results in lengthening of the frame.

61.8.1.1.1 Normal Mode Transmit

The conditions for data transmission from the SSI in Normal mode are:

- SSI Enabled (SSIEN = 1)
- Write data to Transmit Data Register (STX)
- Transmitter Enabled (TE = 1)
- Frame sync active (for continuous clock case)
- Bit clock begins

When the above conditions occur in Normal mode, the next data word is transferred into the Transmit Shift Register (SSI.TXSR) from the Transmit Data Register 0 (SSI.STX0), or from the Transmit FIFO 0 Register, if transmit FIFO 0 is enabled. The new data word is transmitted on arrival of frame-sync preceded by clocks in continuous clock mode. In gated-external mode, data word is transmitted on arrival of frame-sync. In gated-internal mode, data word is transmitted whenever data is available in Tx-FIFO.

If Transmit FIFO 0 is not enabled and the transmit data register empty enable (TDE0_EN) and transmit interrupt enable (TIE) bits are set, transmit interrupt occurs when the word in SSI_STX0 is shifted to Transmit Shift (SSI.TXSR) register for shifting.

If Transmit FIFO 0 is enabled and the transmit fifo empty enable (TFE0_EN) and transmit interrupt enable (TIE) bits are set, transmit interrupt occurs when the number of empty slots in Transmit Fifo 0 exceed or are equal to the selected threshold value i.e.

Transmit Fifo 0 Watermark (TFWM0) value. If transmit FIFO 0 is enabled and filled with data, 15 data words can be transferred before the core must write new data to the SSI.STX0 register.

The STXD port is disabled except during the data transmission period. For a continuous clock, the optional frame sync output and clock outputs are not disabled, even if both receiver and transmitter are disabled.

61.8.1.1.2 Normal Mode Receive

The conditions for data reception from the SSI are:

- SSI enabled (SSIEN = 1)
- Receiver enabled (RE = 1)
- Frame sync active (for continuous clock case)
- Bit clock begins

With the above conditions in Normal mode with a continuous clock, each time the frame sync signal is generated (or detected) a data word is clocked in. With the above conditions and a gated clock, each time the clock begins, a data word is clocked in.

If Receive FIFO 0 is not enabled and Receive Interrupt enable (RIE) and Received Data 0 Ready enable (RDR0_EN) bits are set, receive interrupt occurs when received data word is transferred from the Receive Shift Register (SSI.RXSR) to the Receive Data Register 0 (SSI.SRX0), thus setting the Receive Data Ready 0 (RDR0) flag.

If Receive FIFO 0 is enabled, and Receive Interrupt enable (RIE) and Received Fifo 0 full enable (RFF0_EN) bits are set, receive interrupt occurs when the received data word is transferred to the Receive FIFO 0 and Receive FIFO 0 reaches the selected threshold and results in Receive FIFO Full 0 (RFF0) flag to get set.

The core program has to read the data from the Receive Data Register 0 (SSI.SRX0) (in case Receive FIFO0 is disabled) before a new data word is transferred from the Receive Shift Register (SSI.RXSR), otherwise the Receive Overrun Error 0 (ROE0) bit is set. If receive FIFO 0 is enabled, the Receive Overrun Error 0 (ROE0) bit is set when the Receive FIFO 0 data level reaches the selected threshold and a new data word is ready to be transferred to the Receive FIFO 0.

See the following figure for an illustration of transmitter and receiver timing for an 8-bit word in the first time slot in Normal mode, continuous clock with a late word length frame sync. The Tx Data register is loaded with the data to be transmitted. On arrival of the clock, this data is transferred to the Transmit Shift Register, which gets transmitted on arrival of the frame-sync on the STXD output. Simultaneously, the Receive Shift Register shifts in the received data available on the SRXD input and, at the end of the time slot, this data is transferred to the Rx Data Register.

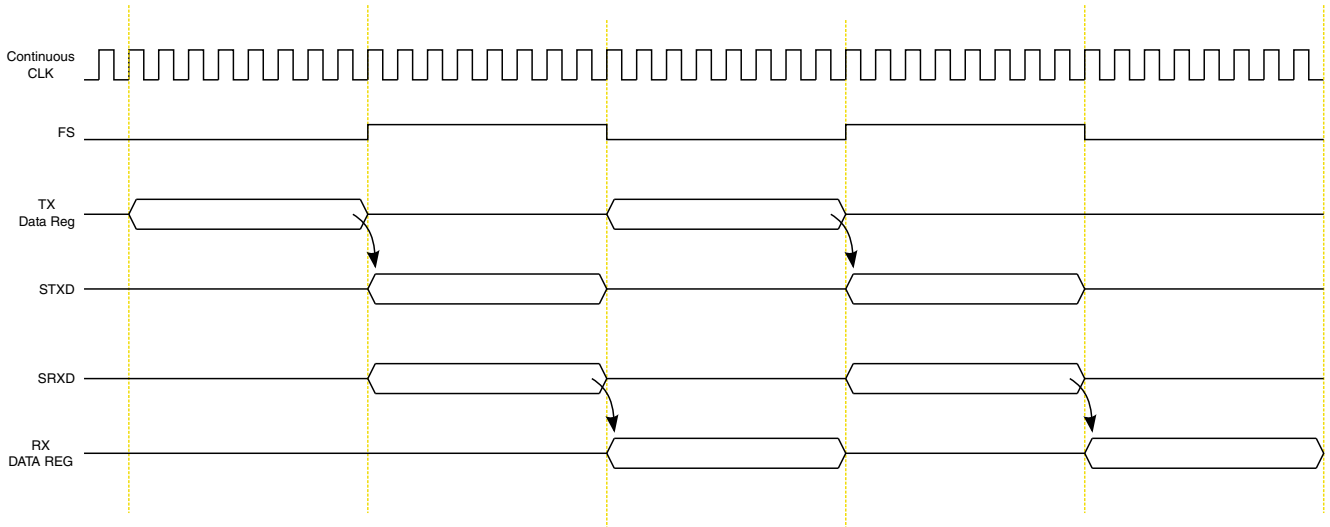


Figure 61-12. Normal Mode Timing - Continuous Clock

The following figure shows a similar case for internal (SSI generates clock) gated clock mode and Figure 61-14 shows a case for external (SSI receives clock) gated clock mode.

NOTE

A pull-down resistor is required in the gated clock case because the clock port is disabled between transmissions.

The Tx Data register is loaded with the data to be transmitted. On arrival of the clock, this data is transferred to the Transmit Shift Register, which gets transmitted on arrival of the frame-sync on the STXD output. Simultaneously, the Receive Shift Register shifts in the received data available on the SRXD input and, at the end of the time slot, this data is transferred to the Rx Data Register. In case of Internal Gated clock mode, the Tx Data line and clock output port are put in the high-impedance state at the end of transmission of the last bit (at the completion of the complete clock cycle), whereas, in External Gated clock mode, the Tx Data line is tri-stated at the last inactive edge of the incoming bit clock (during the last bit in a data word).

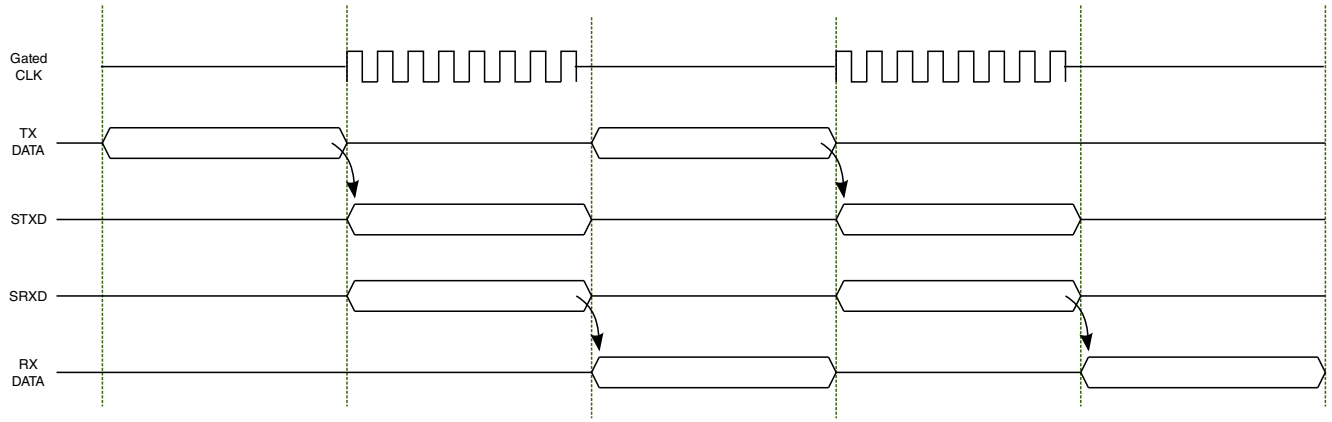


Figure 61-13. Normal Mode Timing - Internal Gated Clock

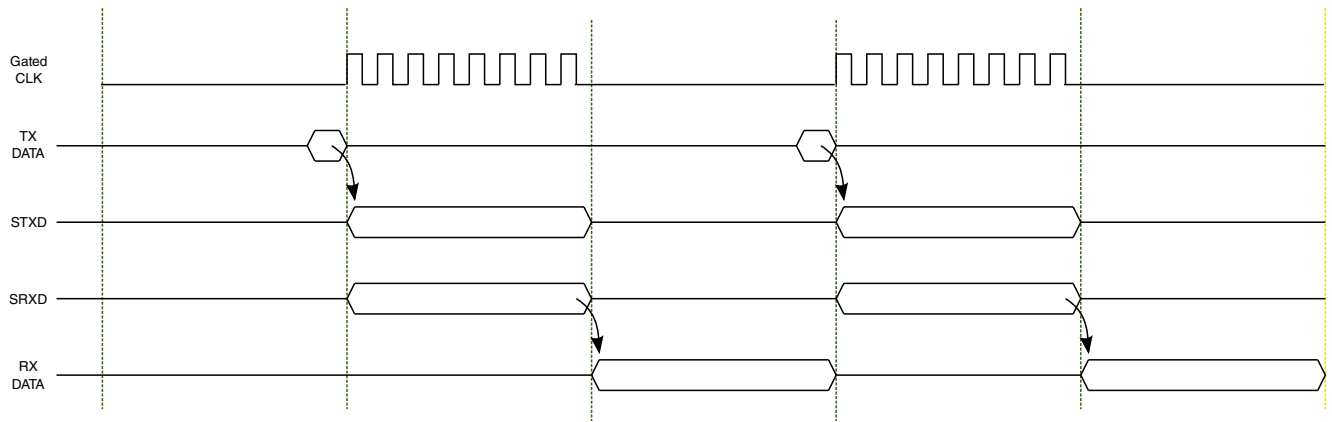


Figure 61-14. Normal Mode Timing - External Gated Clock

61.8.1.2 Network Mode

Network mode is used for creating a Time Division Multiplexed (TDM) network, such as a TDM CODEC network or a network of DSPs.

In Continuous Clock mode, a frame sync occurs at the beginning of each frame. In this mode, the frame is divided into more than one time slot. During each time slot, one data word can be transferred. Each time slot is then assigned to an appropriate CODEC or DSP on the network. The DSP can be a master device that controls its own private network, or a slave device that is connected to an existing TDM network and occupies a few time slots.

The frame sync signal indicates the beginning of a new data frame. Each data frame is divided into time slots and transmission and/or reception of one data word can occur in each time slot (rather than in just the frame sync time slot as in Normal mode). The frame rate dividers, controlled by the DC[4:0] bits, select two to thirty-two time slots per frame. The length of the frame is determined by the following factors:

- The period of the serial bit clock (PSR, PM[7:0] bits for internal clock, or the frequency of the external clock on the STCK port)
- The number of bits per sample (WL[3:0] bits)
- The number of time slots per frame (DC[4:0] bits)

In Network mode, data can be transmitted in any time slot. The distinction of the Network mode is that each time slot is identified with respect to the frame sync (data word time). This time slot identification allows the option of transmitting data during the time slot by writing to the STX registers or ignoring the time slot as determined by SSI.STMSK register bits. The receiver is treated in the same manner and received data is only transferred to the receive data register/fifo if the corresponding time slot is enabled (through SSI.SRMSK).

By utilizing the SSI.STMSK and SSI.SRMSK registers, software only has to service the SSI during valid time slots. This eliminates any overhead associated with unused time slots. See SSI Memory Map/Register Definition for more information on SSI.STMSK and SSI.SRMSK.

In the Two-Channel mode of operation, the second set of Transmit and Receive FIFOs and Data Registers are used to create two separate channels. These channels are completely independent, with a their own set of Core interrupts and DMA requests, which are identical to the ones available for the default channel. In this mode, data is transmitted/received in enabled time slots alternately from/to FIFO 0 and FIFO 1, starting from FIFO 0. The first data word is taken from FIFO 0 and transmitted in the first enabled time slot and subsequently, data is loaded from FIFO 1 and FIFO 0 alternately and transmitted. Similarly, the first received data is sent to FIFO 0 and subsequent data is sent to FIFO 1 and FIFO 0 alternately. Time slots can be selected through the Transmit and Receive Time Slot Mask registers (SSI.STMSK and SSI.SRMSK). For using this mode of operation, the TCH_EN bit (SCR[8]) needs to be set.

61.8.1.2.1 Network Mode Transmit

The transmit portion of SSI is enabled when the SSIEN and the TE bits in the SSI.SCR are both set. However, for continuous clock, when the TE bit is set, the transmitter is enabled only after detection of a new frame sync (transmission starts from the next frame boundary).

Normal start-up sequence for transmission is to perform the following:

1. Enable Network Mode.
2. Enable SSI
3. Write the data to be transmitted to the SSI.STX register. This clears the TDE flag
4. Set the TE bit to enable the transmitter on the next frame boundary (for continuous clock case).
5. Enable transmit interrupts.

(Alternatively, the programmer may decide not to transmit in a time slot by configuring the STMSK.) TDE flag is set as data is shifted from SSI.STX register to TXSR, but the STXD port remains disabled during the time slots. When the next frame sync is detected or generated (continuous clock), the data word in TXSR and is shifted out (transmitted). When the SSI.STX register is empty, the TDE bit is set, which causes a transmitter interrupt (in case the FIFO is disabled) to be sent if the TIE bit is set. Software can poll the TDE bit or use interrupts to reload the STX register with new data for the next time slot. Failing to reload the SSI.STX register before the TXSR is finished shifting (empty) causes a transmitter underrun and the TUE error bit is set. In case the FIFO is enabled, the TFE flag is set in accordance with the watermark setting and this flag causes the transmitter interrupt to occur.

The operation of clearing the TE bit disables the transmitter after completion of transmission of the current frame. Setting the TE bit enables transmission from the next frame. During that time the STXD port is disabled. The TE bit should be cleared after the TDE bit is set to ensure that all pending data is transmitted.

To summarize, the Network mode transmitter generates interrupts every enabled time slot (when FIFO is disabled) and requires the core program to respond to each enabled time slot. These responses from the core are one of the following:

- Write data in data register to enable transmission in the next time slot.
- Configure the time slot register to disable transmission in the next time slot (unless time slot is already masked by SSI.STMSK register bit).
- Do nothing-transmit underrun occurs at the beginning of the next time slot and the previous data is re-transmitted.

In the Two-Channel mode of operation, both the channels (Data Registers, FIFOs, Interrupts and DMA requests) operate in the same manner, as described above. The only difference in case of the second channel is that the Interrupts related to this channel are generated only in case this mode of operation is selected (TDE1 is low by default).

61.8.1.2.2 Network Mode Receive

The receiver portion of the SSI is enabled when both the SSIEN and the RE bits in the SSI.SCR are set. However, the receive enable only takes place during that time slot if RE is enabled before the second to last bit of the word. If the RE bit is cleared, the receiver is disabled at the end of the current frame.

SSI is capable of finding the start of the next frame automatically. When the word is completely received, it is transferred to the SRX register, which sets the RDR bit (Receive Data Ready). Setting the RDR bit causes a receive interrupt to occur if the receiver interrupt is enabled (the RIE bit is set) and (Receive data ready enable) RDR_EN bit is set. The second data word (second time slot in the frame), begins shifting in immediately after the transfer of the first data word to the SSI.SRX register. The core program has to read the data from the Receive Data Register (which clears RDR) before the second data word is completely received (ready to transfer to RX data register) or a receive overrun error occurs (the ROE bit is set).

An interrupt can occur after the reception of each enabled data word or the programmer can poll the RDR flag. The core program response can be one of the following:

- Read RX and use the data.
- Read RX and ignore the data.
- Do nothing-the receiver overrun exception occurs at the end of the current time slot.

NOTE

For a continuous clock, the optional frame sync output and clock output signals are not affected, even if the transmitter or receiver is disabled. TE and RE do not disable the bit clock or the frame sync generation. To disable the bit clock and the frame sync generation, the SSIEN bit in the SSI.SCR can be cleared or TFR_CLK_DIS/RFR_CLK_DIS bits can be set, or the port control logic external to the SSI (for example, in the IOMUXC) can be re configured.

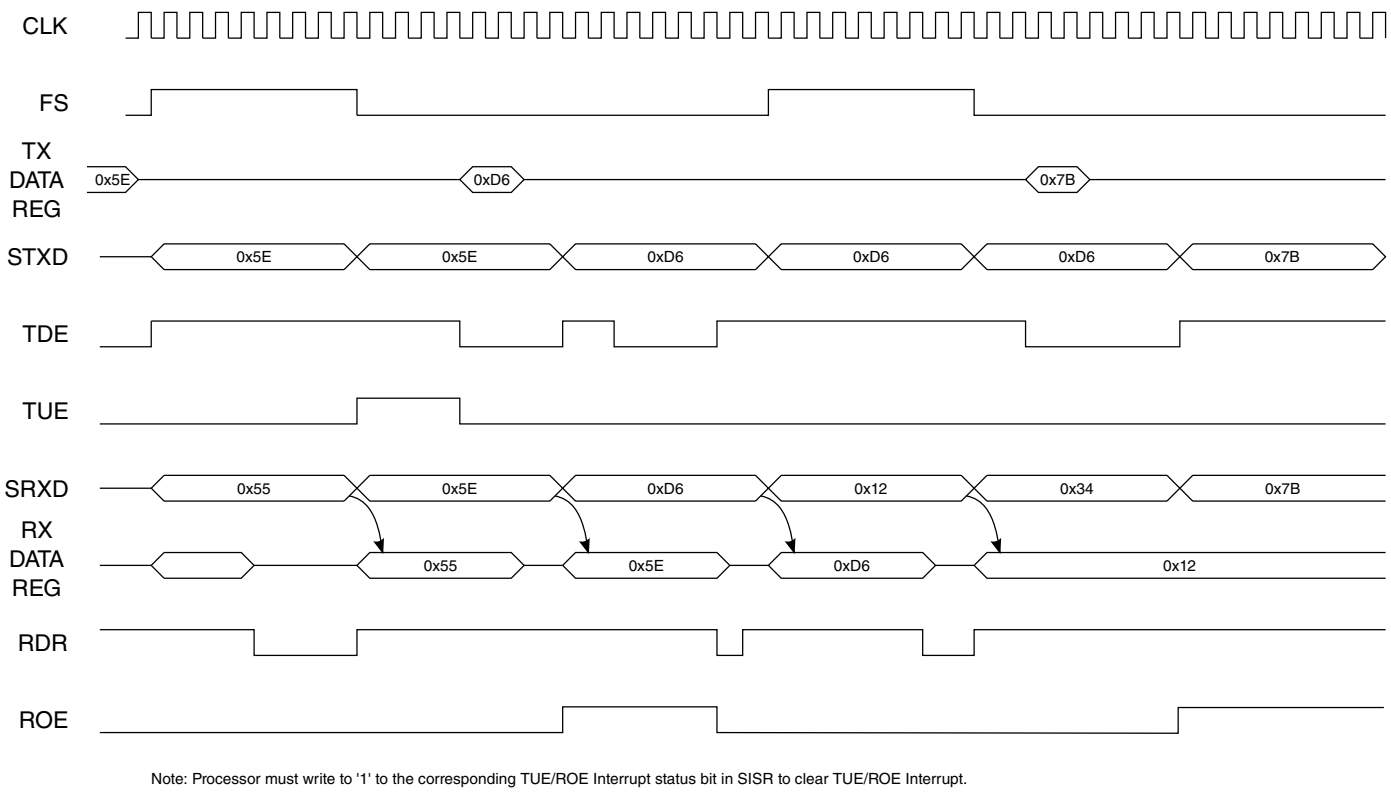
In the Two-Channel mode of operation, both the channels (Data Registers, FIFOs, Interrupts and DMA requests) operate in the same manner, as described above. The only difference in case of the second channel is that the Interrupts related to this channel are generated only in case this mode of operation is selected.

The transmitter and receiver timing for an 8-bit word with continuous clock, FIFO disabled, three words per frame sync in Network mode is shown in the figure below).

NOTE

The transmitter repeats the value 0x5E because of an underrun condition

For the receive section, data received on the SRXD pin gets transferred to the Rx Data register at the end of each time slot. If the FIFO is disabled, the RDR flag gets set and causes a receiver interrupt if RE, RIE and RDR_EN bits are set. If the FIFO is enabled, then the RFF flag is used for interrupt generation (this flag is set in accordance with the watermark settings). Here all time slots are enabled. The receive data ready flag is set after reception of the first data (0x55). Since the flag is not cleared (Rx Data Register is not read by core), the Receive Overrun Error (ROE) flag is set on reception of the next data (0x5E). ROE flag is cleared on writing '1' to the corresponding interrupt status bit in SSI Status Register.



Note: Processor must write to '1' to the corresponding TUE/ROE Interrupt status bit in SISR to clear TUE/ROE Interrupt.

Figure 61-15. Network Mode Timing - Continuous Clock

61.8.1.3 Gated Clock Mode

Gated Clock mode is often used to connect to SPI-type interfaces on Micro controller Units (MCUs) or external peripheral chips. In Gated Clock mode, the presence of the clock indicates that valid data is on the STXD or SRXD ports.

For this reason, no frame sync is needed in this mode. Once transmission of data has completed, the clock is pulled to the inactive state. Gated clocks are allowed for both the transmit and receive sections with either internal or external clock in Normal mode. Gated clocks are not allowed in Network mode. See Table 61-2 ("Clock Pin Configurations") for SSI configuration for gated-mode operation.

The clock runs when the TE bit and/or the RE bit are appropriately enabled. For the case of internally generated clock, all internal bit clocks, word clocks, and frame clocks continue to operate. When a valid time slot occurs (such as the first time slot in Normal mode), the internal bit clock is enabled onto the appropriate clock port. This allows data to be transferred out in periodic intervals in Gated Clock mode. With an external clock, the SSI waits for a clock signal to be received. Once the clock begins, valid data is shifted in. Care should be taken to clear all DC bits (0x00000) when SSI is used in Gated mode. In gated mode of operation the TFS, RFS, TLS, RLS, TFRC and RFRC bits of SSI.AISR register are not generated.

For Gated clock operated in external clock mode, a proper clock signalling must be applied to the SSI STCK in order for it to function properly. When TSCKP is 0, CLK_IST value should be 1. When TSCKP is 1, CLK_IST value should be 0. If the SSI uses rising edge transition to clock data (TSCKP=0) and the falling edge transition to latch data (RSCKP=0), the clock must be in an active low state when idle. If the SSI uses falling edge transition to clock data (TSCKP=1) and the rising edge transition to latch data (RSCKP=1), the clock must be in a active high state when idle. The figures below illustrate the different edge clocking/latching.

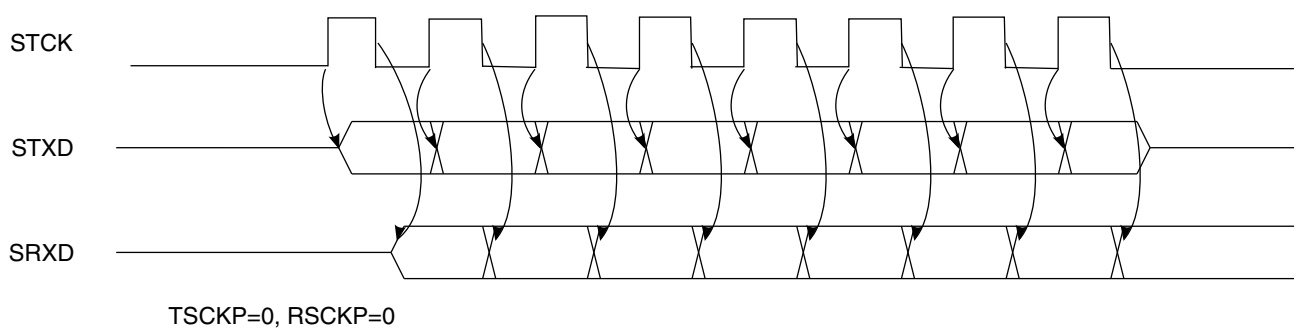


Figure 61-16. Internal Gated Mode Timing - Rising Edge Clocking / Falling Edge Latching

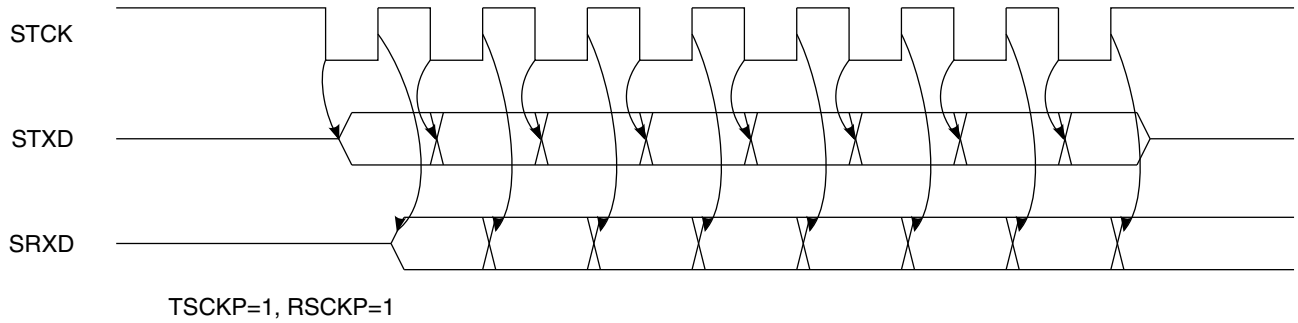


Figure 61-17. Internal Gated Mode Timing - Falling Edge Clocking / Rising Edge Latching

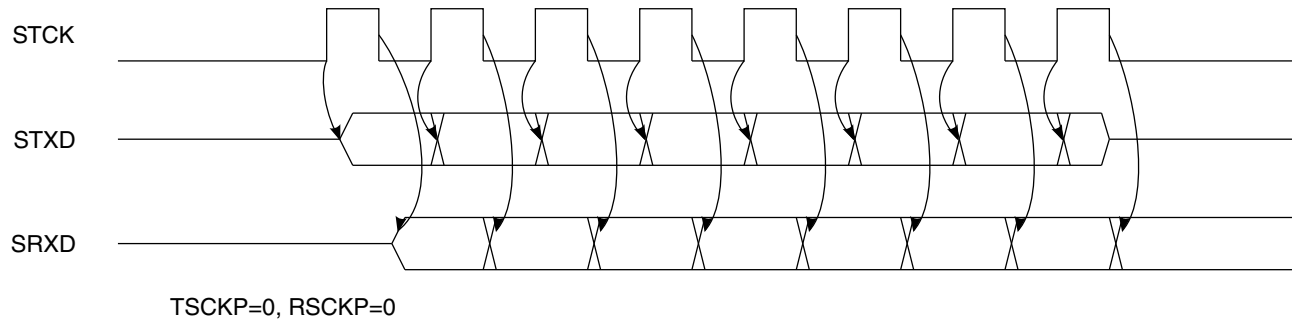


Figure 61-18. External Gated Mode Timing - Rising Edge Clocking / Falling Edge Latching

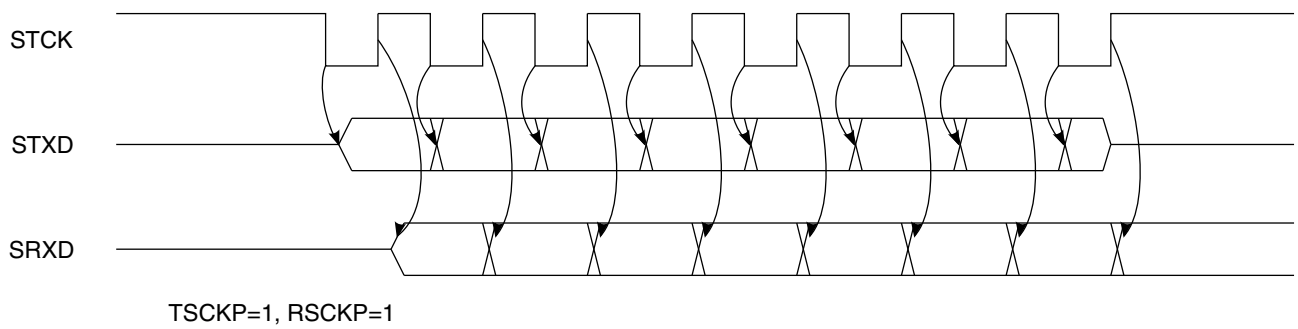


Figure 61-19. External Gated Mode Timing - Falling Edge clocking / Rising Edge Latching

The bit clock ports must be kept free of timing glitches. If a single glitch occurs, all ensuing transfers will be out of synchronization.

In case of External Gated Mode, even though the Tx Data line is put in the high-impedance state at the last non-active edge of the bit clock, the round trip delay should be sufficient to take care of hold time requirements at the external receiver.

61.8.1.4 I2S Mode

The SSI is compliant to the Inter-IC Sound (I2S) bus specification from Philips Semiconductors (February 1986, Revised June 5, 1996). For more information on I2S, refer to the latest version of NXP Semiconductor's I2S specification.

See the following figure for an illustration of the basic I2S protocol timing.

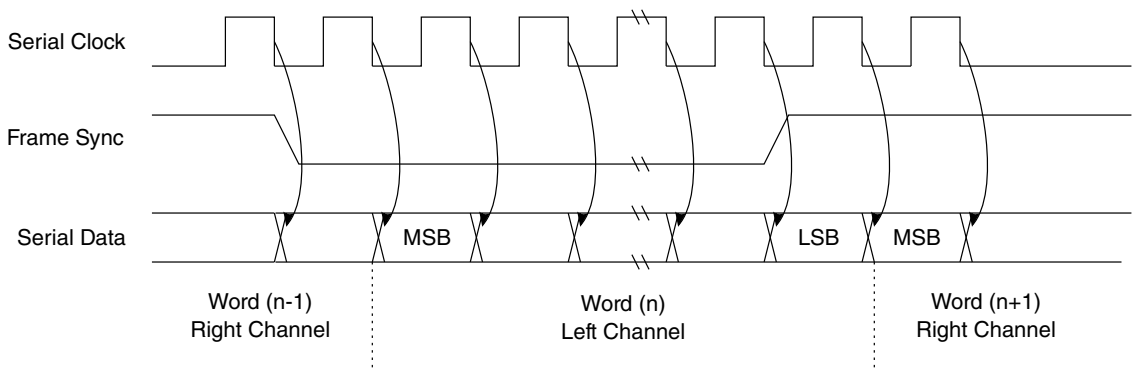


Figure 61-20. I2S Mode Timing - Serial Clock, Frame Sync and Serial Data

Select I2S mode using the options listed in the table below.

Table 61-5. I2S Mode Selection

I2S_MODE[1]	I2S_MODE[0]	Mode Type
0	0	Normal mode
0	1	I2S master mode
1	0	I2S slave mode
1	1	Normal mode

In normal mode operation, no register bits are forced to any particular state internally and the user can program the SSI to work in any operating condition.

When I2S modes are entered (I2S master (01) or I2S slave (10)), the following settings are recommended:

- Sync mode (SSI_SCR[4] =1)
- Tx shift direction: MSB transmitted first (SSI_STCR[4]=0)
- Rx shift direction: MSB received first (SSI_SRCR[4]=0)
- Tx data clocked at falling edge of the clock (SSI_STCR[3]=1)
- Rx data latched at rising edge of the clock (SSI_SRCR[3]=1)
- Tx frame sync active low (SSI_STCR[2]=1)
- Rx frame sync active low (SSI_SRCR[2]=1)

- Tx frame sync initiated one bit before data is transmitted (SSI_STCR[0]=1)
- Rx frame sync initiated one bit before data is received (SSI_SRCR[0]=1)
- TX Frame Rate should be 2 (SSI_STCCR[12:8] = 1)
- RX Frame Rate should be 2 (SSI_SRCCR[12:8] = 1)

In I2S master mode (SSI_SCR[6:5]=01), the following additional settings are recommended:

- TXDIR bit (SSI_STCR[5]) set to 1 to select internal generated bit clock
- TFDIR bit (SSI_STCR[6]) set to 1 to select internal generated frame sync

In I2S master mode(SSI.SCR[6:5]=01), the following settings are internally overridden by the hardware:

- Network mode is selected (SSI_SCR[3]=1)
- Tx frame sync length set to one-word-long-frame (SSI_STCR[1]=0)
- Rx frame sync length set to one-word-long-frame (SSI_SRCR[1]=0)
- Tx shifting with respect to bit 0 of TXSR (SSI.STCR[9]=1)
- Rx shifting with respect to bit 0 of RXSR (SSI.SRCR[9]=1)

The user needs to set the following control bits to configure the bit clock and frame sync:

- PM (SSI_STCCR[7:0])
- PSR (SSI_STCCR[17])
- DIV2(SSI_STCCR[18])
- WL (SSI_STCCR[16:13])
- DC (SSI_STCCR[12:8])

The word length is fixed to 32 in I2S Master mode and the WL bits determine the number of bits that will contain valid data (out of the 32 transmitted/received bits in each channel).

In I2S slave mode(SSI_SCR[6:5]=10), the following additional settings are recommended:

- TXDIR bit(SSI_STCR[5]) set to 0 to select external generated bit clock
- TFDIR bit(SSI_STCR[6]) set to 0 to select external generated frame sync

In I2S slave mode(SSI_SCR[6:5]=10), the following settings are internally overridden by the hardware:

- Normal mode is selected (SSI_SCR[3]=0)
- Tx frame sync length set to one-bit-long-frame (SSI_STCR[1]=1)
- Rx frame sync length set to one-bit-long-frame (SSI_SRCR[1]=1)
- Tx shifting with respect to bit 0 of TXSR (SSI_STCR[9]=1)
- Rx shifting with respect to bit 0 of RXSR (SSI_SRCR[9]=1)

The user needs to set the following control bits to configure the data transmission:

- WL (SSI_STCCR[16:13])
- DC (SSI_STCCR[12:8])

The word length is variable in I2S slave mode and the WL bits determine the number of bits that will contain valid data. The actual word length is determined by the external CODEC. The external I2S Master still sends frame sync according to the I2S protocol (early, word wide and active low), the SSI internally operates so that each frame sync transition is the start of a new frame (the WL bits determine the number of bits to be transmitted/received). After one data word has been transferred, the SSI waits for the next frame sync transition to start operation in the next time slot. Transmit (STMSK) and receive (SRMSK) mask bits should not be used in I2S Slave mode of operation. Masking is supported only for network mode of operation.

61.8.1.5 AC97 Mode

In AC97 mode of operation, the SSI transmits a 16-bit Tag Slot at the start of a frame and the rest of the slots (in that frame) are all 20-bits wide.

The same sequence is followed while receiving data. Refer to the AC97 specification for details regarding transmit and receive sequences and data formats.

NOTE

The Audio Codec specification released in 1997 [AC '97] defines the Architecture and Digital Interface, specifically designed for implementing audio and modem I/O functionality in personal computers. Companion specifications include the Modem Codec [MC '97], and the combined Audio/Modem Codec standard [AMC '97]. The current version of AC '97 was produced in 2002. The AC-97 specification defines a recommended 48-pin QFP IC package.

Note that the SSI only has one RxDATA pin so the SSI can only support one codec. Secondary codecs are not supported.

When AC97 mode is enabled, the following settings are internally overridden by the hardware. The programmed register values are not changed by entering AC97 mode but they no longer apply to the block's operation. Writing to the programmed register fields will update their values; these updates can be seen by reading back the register fields. However, these settings will not take effect until AC97 mode is turned off.

The register bits within the bracket are the equivalent settings:

- Sync mode is entered (SSI.SCR[4] =1)
- Network mode is selected (SSI.SCR[3]=1)
- Tx shift direction is MSB transmitted first (SSI.STCR[4]=0)
- Rx shift direction is MSB received first (SSI.SRCR[4]=0)
- Tx data is clocked at rising edge of the clock (SSI.STCR[3]=0)
- Rx data is latched at falling edge of the clock (SSI.SRCR[3]=0)
- Tx frame sync is active high (SSI.STCR[2]=0)
- Rx frame sync is active high (SSI.SRCR[2]=0)
- Tx frame sync length is one-word-long-frame (SSI.STCR[1]=0)
- Rx frame sync length is one-word-long-frame (SSI.SRCR[1]=0)
- Tx frame sync initiated one bit before data is transmitted (SSI.STCR[0]=1)
- Rx frame sync initiated one bit before data is received (SSI.SRCR[0]=1)
- Tx shifting with respect to bit 0 of TXSR (SSI.STCR[9]=1)
- Rx shifting with respect to bit 0 of RXSR (SSI.SRCR[9]=1)
- Tx FIFO is enabled (SSI.STCR[7]=1)
- Rx FIFO is enabled (SSI.SRCR[7]=1)
- TFDIR bit (SSI.STCR[6]) is forced to 1 internally to select internal generated frame sync
- TXDIR bit (SSI.STCR[5]) is forced to 0 internally to select external generated bit clock

Any alteration of these bits individually will not affect the operational conditions of the SSI unless AC97 mode is deselected.

Hence, the only control bits needed to be set by the user to configure the data transmission/reception are the WL (SSI.STCCR[16:13]) and DC (SSI.STCCR[12:8]) bits. In AC97 mode, the WL bits can only legally take the values corresponding to 16-bit (truncated data) or 20-bit time slots. In case WL bits are set to select 16-bit time slots, the SSI pads the transmit data (four least significant bits) with zeros and while receiving, stores only the most significant 16 bits in the Rx FIFO.

Follow the sequence for programming the SSI to work in AC97 mode:

1. Program the WL bits to a value corresponding to either 16 or 20 bits. The WL bit setting is only for the data portion of the AC97 frame (Slots #3 through #12). The Tag slot (Slot #0) is always 16 bits wide and the Command Address and Command Data slots (Slots #1 and #2) are always 20 bits wide.
2. Select the number of time slots by programming the DC bits. For AC97 operation, DC bits should be set to a value of '0xC', resulting in 13 time slots per frame.
3. Write data to be transmitted, in Tx FIFO 0 (through Tx Data Register 0) and Tx FIFO 1 while using Two-Channel Mode (TCH_EN = 1).
4. Program the FV, TIF, RD, WR and FRDIV bits in SSI.SACNT register

5. Update the contents of SSI.SACADD, SSI.SACDAT and SSI.SATAG (for Fixed mode only) registers
6. Enable the AC97 mode of operation (AC97EN bit in SSI.SACNT register)

Once the SSI starts transmitting and receiving data (after being configured in AC97 mode), the programmer needs to service the interrupts, as and when they are raised (updates to command address/data or tag registers, reading of received data and writing more data for transmission). Further details regarding fixed and variable mode implementation are provided in the following sections.

While using AC97 in Two-Channel Mode (TCH_EN=1), it is recommended that the received tag is not stored in the Rx FIFO (TIF=0). In case the programmer needs to update the SSI.SATAG register and also issue a RD/WR command (in a single frame), it is recommended that the SSI.SATAG register be updated prior to issuing a RD/WR command.

61.8.1.5.1 AC97 Fixed Mode (SSI.SACNT[1]=0)

In fixed mode of operation, SSI transmits in accordance with the AC97 Frame Rate Divider bits (i.e. FRDIV in SACNT) which decides the number of frames for which the SSI should be idle, after operating for one frame.

In a valid frame, TAG Value (written by Core) will be transmitted in Slot #0, Command Address will be transmitted in Slot #1 in case of RD/WR Command, and Command Data will be transmitted in Slot #2 in case of a WR Command. The data from TX-FIFO is transmitted in Slot #3 - Slot #12 depending on the valid slots indicated by the TAG value.

While receiving, bit 15 of the TAG Value (Slot #0) is checked to see if the CODEC is ready. If this bit is set, the frame is received. The received TAG provides the information about Slots containing valid data. The the corresponding TAG bit is valid, the Command Address (Slot #1) and Command Data (Slot #2) values are stored in the corresponding registers. The received data (Slot #3 - Slot #12) is then stored in the Rx-FIFO (for valid slots).

61.8.1.5.2 AC97 Variable Mode (SSI.SACNT[1]=1)

In Variable Mode, the transmit slots which should contain data in the current frame are determined by SLOTREQ bits received in the previous frame. While receiving, if the CODEC is ready, the frame is received and the SLOTREQ bits (contained in Slot #1) are stored for scheduling transmission in the next frame.

The SSI.SACCST, SSI.SACCEN and SSI.SACCDIS registers helps in determining which transmit slots are active. This information is used to ensure that SSI does not transmit data for powered-down/inactive channels.

61.8.2 External Frame and Clock Operation

When applying external frame sync and clock signals to SSI, there should be at least 4-bit clock cycles between the enabling of the transmit or receive section and the rising edge of the corresponding frame sync signal.

The transition of STFS or SRFS should be synchronized with the rising edge of external clock signal, STCK or SRCK.

61.8.2.1 Data Alignment Formats Supported

The SSI supports three data formats in order to provide flexibility with handling data. These formats dictate how data is written to (and read from) the data registers. Therefore, data can appear in different places in SSI.STX0/1 and SSI.SRX0/1 based on the data format and the number of bits per word. Independent data formats are supported for both the transmitter and receiver (that is, the transmitter and receiver can use different data formats).

The supported data formats are:

- MSB alignment
- LSB alignment
  - Zero-extended (receive data only)
  - Sign-extended (receive data only)

With MSB alignment, the most significant byte is bits 31 through 24 of the data register if the word length is larger than or equal to 16 bits. If the word length is less than 16 bits and MSB alignment is chosen, the most significant byte is bits 15 through 8. With LSB alignment, the least significant byte is bits 7 through 0. Data alignment is controlled by the TXBIT0 bit in the SSI.STCR and the RXBIT0 bit in the SSI.SRCR. See the table below for the bit assignment for all the data formats supported by the SSI.

Table 61-6. Data Alignment

Format	Bit Number																															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
8-bit LSB Aligned																									7	6	5	4	3	2	1	0
8-bit MSB Aligned																	7	6	5	4	3	2	1	0								
10-bit LSB Aligned																							9	8	7	6	5	4	3	2	1	0
10-bit MSB Aligned																	9	8	7	6	5	4	3	2	1	0						

Table continues on the next page...

Table 61-6. Data Alignment (continued)

12-bit LSB Aligned																					11	10	9	8	7	6	5	4	3	2	1	0
12-bit MSB Aligned																	11	10	9	8	7	6	5	4	3	2	1	0				
16-bit LSB Aligned																	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
16-bit MSB Aligned	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																
18-bit LSB Aligned															17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
18-bit MSB Aligned	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0														
20-bit LSB Aligned													19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
20-bit MSB Aligned	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0												
22-bit LSB Aligned											21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
22-bit MSB Aligned	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0										
24-bit LSB Aligned									23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24-bit MSB Aligned	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								

In addition, receive data can either be zero-extended or sign-extended if LSB alignment is selected. With zero-extension, all bits above the most significant bit are 0's. This format is useful when data is stored in a pure integer format. With sign-extension, all bits above the most significant bit are equal to the most significant bit. This format is useful when data is stored in a fixed-point integer format (which implies fractional values). Receive data extension is controlled by the RXEXT bit in the SSI.SRCR. Transmit data used with LSB alignment has no concept of sign/zero-extension. Unused bits above the most significant bit are simply ignored.

When configured in I2S or AC97 mode, the SSI forces the selection of LSB alignment. However, RXEXT still permits a choice between zero-extension and sign-extension.

See SSI for more details on the relevant bits in the SSI.STCR and SSI.SRCR registers.

61.8.3 SSI Architecture

The Synchronous Serial Interface (SSI) is connected to chip pads through the Digital Audio Mux (AUDMUX) block. The AUDMUX can be configured to connect the SSI to the chip pads in various ways.

Refer to Figure 61-1 for a block diagram of the SSI.

61.8.4 SSI Clocking

The SSI uses the following clocks:

- Bit clock - Used to serially clock the data bits in and out of the SSI port. This clock is either generated internally (from SSI's sys clock) or taken from external clock source (through the Tx/Rx clock ports).
- Word clock - Used to count the number of data bits per word (8, 10, 12, 16, 18, 20, 22 or 24 bits). This clock is generated internally from the bit clock.
- Frame clock (Frame Sync) - Used to count the number of words in a frame. This signal can be generated internally from the bit clock, or taken from external source (from the Tx/Rx frame sync ports).
- Network clock -In master mode, this is an integer multiple of frame clock. This is oversampling clock. It is used in cases when SSI has to provide the clock.

Care should be taken to ensure that the bit clock frequency (either internally generated by dividing the SSI's sys clock or sourced from external device through Tx/Rx clock ports) is never greater than 1/5 of the ipg_clk (from CCM) frequency.

In Normal mode (SCR[6:5]=00), the bit clock, used to serially clock the data, is visible on the Serial Transmit Clock (STCK) and Serial Receive Clock (SRCK) ports. The word clock is an internal clock used to determine when transmission of an 8, 10, 12, 16, 18, 20, 22 or 24 bit word has completed. The word clock in turn then clocks the frame clock, which counts the number of words in the frame. The frame clock can be viewed on the STFS and SRFS frame sync ports, because a frame sync is generated after the correct number of words in the frame have passed. In master and synchronous mode, the unused port SRCK is used as network clock (oversampling clock) enabled by the SCR register bit 15, SYS_CLK_EN. This network clock is an oversampling clock of the frame sync clock (STFS). In this mode, the word length (WL), Prescaler Range (PSR), Prescaler Modulus (PM) and Frame rate (DC) selects the ratio of network clock to sampling clock STFS. In case of I2S mode, the network clock (oversampling clock) can be made available on this port if the SYS_CLK_EN bit is set. The relationship between the clocks and the dividers is shown in the figure below ("SSI Clocking"). The bit clock can be received from an SSI clock port or can be generated from the network clock through a divider, as shown in Figure 61-22 ("SSI Transmit Clock Generator Block Diagram").

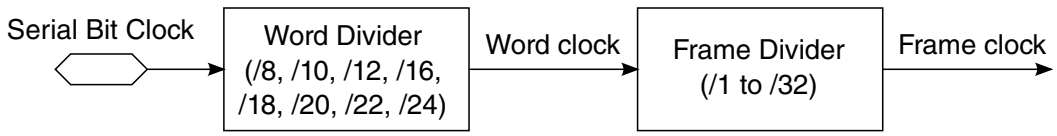


Figure 61-21. SSI Clocking

61.8.4.1 SSI Clock and Frame Sync Generation

Data clock and frame sync signals can be generated internally, or can be obtained from external sources. If internally generated, the SSI clock generator is used to derive bit clock and frame sync signals from the SSI's sys clock. The SSI clock generator consists of a selectable, fixed prescaler and a programmable prescaler for bit rate clock generation.

In Gated Clock mode, the data clock is valid only when data is being transmitted. Otherwise the clock port is pulled to the inactive state. A programmable frame rate divider and a word length divider are used for frame rate sync signal generation.

The following figure shows a block diagram of the clock generator for the transmit section. The serial bit clock can be internal or external, depending on the Transmit Direction (TXDIR) bit in the SSI Transmit Configuration Register (SSI.STCR). The receive section contains an equivalent clock generator circuit.

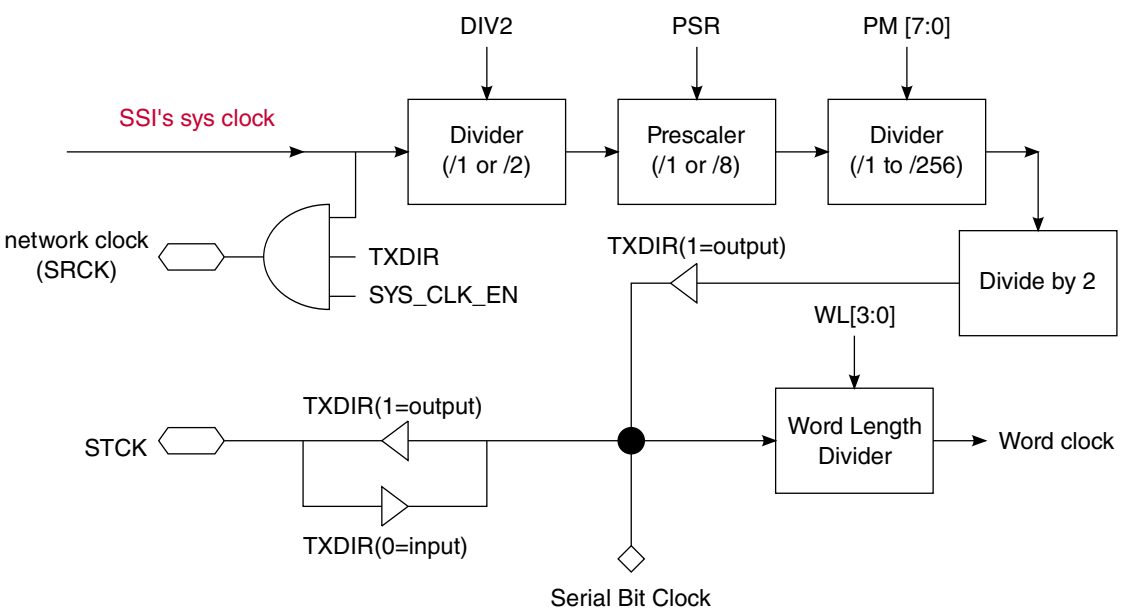


Figure 61-22. SSI Transmit Clock Generator Block Diagram

The figure below shows the Frame Sync Generator block for the transmit section. When internally generated, both receive and transmit frame sync are generated from the word clock and are defined by the Frame Rate Divider (DC[4:0]) bits and the Word Length (WL[3:0]) bits of the SSI Transmit Clock Control Register (SSI.STCCR). The receive section contains an equivalent circuit for the Frame Sync Generator.

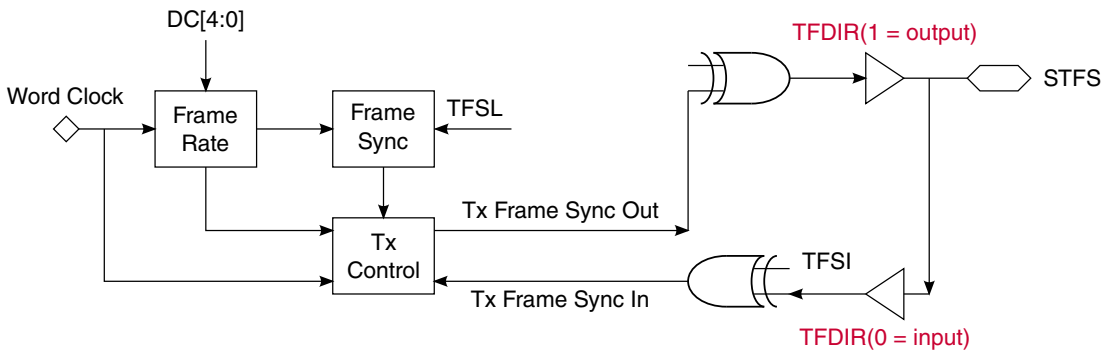


Figure 61-23. SSI Transmit Frame Sync Generator Block Diagram

61.8.4.2 DIV2, PSR and PM Bit Description

The bit clock frequency can be calculated from the SSI's sys clock using the equation in the following figure.

NOTE

You must ensure that the bit-clock frequency must be never greater than 1/5 of the peripheral clock frequency. The oversampling clock frequency can go up to peripheral clock frequency. Bits DIV2, PSR and PM should not be all set to zero at the same time.

$$f_{INT\_BIT\_CLK} = f_{SSI's\ sys\ clock} / [(DIV2 + 1) \times (7 \times PSR + 1) \times (PM + 1) \times 2]$$

where PM = PM[7:0]

$$f_{FRAME\_SYN\_CLK} = (F_{INT\_BIT\_CLK}) / [(DC + 1) \times WL]$$

where DC = DC[4:0] and WL = 8, 10, 12, 16, 18, 20, 22, 24

Figure 61-24. SSI Bit Clock Equation

For example, if the SSI's sys clock is 12.288 MHz, in 8-bit word Normal mode with DC[4:0] set to 0(00000), PM[7:0] set to 47 (0010 1111), the PSR bit cleared, DIV2 bit set to 1, a bit clock rate of 12.288 Mhz / [1 x 4 x 48] = 64 kHz is generated. Since the 8-bit word rate is equal to one (i.e. normal mode), the sampling rate (FS rate) would then be 64 kHz / [1 * 8] = 8 kHz.

In the next example, SSI's sys clock is 11.2896 Mhz. A 16-bit word Network mode with DC[4:0] set to 1 (00001), PM[7:0] set to 3 (0000 0011), the PSR bit is set to 0, DIV2 bit set to 0, and a 11.2896 MHz oversampling clock, a bit clock rate of 11.2896 Mhz / [1 x 2 x 4] = 1.4112 MHz is generated. Since the 16-bit word rate is equal to two, the sampling rate (FS rate) would be 1.4112 MHz / [2 * 16] = 44.1 kHz.

The table below shows programming examples for the clock dividers in the CCM and the SSI to support various bit clock (STCK) frequencies.

Table 61-7. SSI Bit Clock and Frame Rate as a Function of PSR, PM, and DIV2

Bits/ Word	Words / Frame	Ideal Frame Rate (kHz)	PLL Freq (Mhz)	SSIDIV (in CCM)	SSI's sys clock Freq (Mhz)	DIV2	PSR	PM	WL	DC	Actual Serial bit clock Freq (kHz) STCK	Target Serial bit clock Freq (kHz) STCK	Error (Hz)
16	1	8	688.128	56	12.288	0	0	47	7	0	128	128	0
16	2	8	688.128	56	12.288	0	0	23	7	1	256	256	0
16	4	8	688.128	56	12.288	0	0	11	7	3	512	512	0
16	1	12	688.128	56	12.288	0	0	31	7	0	192	192	0
16	2	12	688.128	56	12.288	0	0	15	7	1	384	384	0
16	4	12	688.128	56	12.288	0	0	7	7	3	768	768	0
16	1	16	688.128	56	12.288	0	0	23	7	0	256	256	0
16	2	16	688.128	56	12.288	0	0	11	7	1	512	512	0
16	4	16	688.128	56	12.288	0	0	5	7	3	1024	1024	0
16	1	24	688.128	56	12.288	0	0	15	7	0	384	384	0
16	2	24	688.128	56	12.288	0	0	7	7	1	768	768	0
16	4	24	688.128	56	12.288	0	0	3	7	3	1536	1536	0
16	1	32	688.128	56	12.288	0	0	11	7	0	512	512	0
16	2	32	688.128	56	12.288	0	0	5	7	1	1024	1024	0
16	4	32	688.128	56	12.288	0	0	2	7	3	2048	2048	0
16	1	48	688.128	56	12.288	0	0	15	7	0	768	768	0
16	2	48	688.128	56	12.288	0	0	3	7	1	1536	1536	0
16	4	48	688.128	56	12.288	0	0	1	7	3	3072	3072	0
16	1	11.025	632.2176	56	11.2896	0	0	31	7	0	176.4	176.4	0

Table continues on the next page...

Table 61-7. SSI Bit Clock and Frame Rate as a Function of PSR, PM, and DIV2 (continued)

Bits/ Word	Words / Frame	Ideal Frame Rate (kHz)	PLL Freq (Mhz)	SSIDIV (in CCM)	SSI's sys clock Freq (Mhz)	DIV2	PSR	PM	WL	DC	Actual Serial bit clock Freq (kHz) STCK	Target Serial bit clock Freq (kHz) STCK	Error (Hz)
16	2	11.025	632.2176	56	11.2896	0	0	15	7	1	352.8	352.8	0
16	4	11.025	632.2176	56	11.2896	0	0	7	7	3	705.6	705.6	0
16	1	22.05	632.2176	56	11.2896	0	0	15	7	0	352.8	352.8	0
16	2	22.05	632.2176	56	11.2896	0	0	7	7	1	705.6	705.6	0
16	4	22.05	632.2176	56	11.2896	0	0	3	7	3	1411.2	1411.2	0
16	1	44.1	632.2176	56	11.2896	0	0	7	7	0	705.6	705.6	0
16	2	44.1	632.2176	56	11.2896	0	0	3	7	1	1411.2	1411.2	0
16	4	44.1	632.2176	56	11.2896	0	0	1	7	3	2822.4	2822.4	0

NOTE

The table above describes how various frame rates can be achieved with the PLLs supplying a frequency of 688.128 MHz and 633.2176 MHz (with WL and DC settings as shown). These clocks are recommended as convenient starting points but the system allows for other input clock frequencies as well.

Table 61-7 shows programming of the CCM and SSI dividers in order to generate the appropriate network clock and serial bit clock frequencies for various sampling rates. In these examples, the master mode is selected either by setting I2S master bit (SCR[6:5]=01) or individually programming the SSI in network, synchronous, transmit internal mode (the table specifically illustrates the I2S mode frequencies/sample rates). The network clock is oversampling clock.

Note that the I2S master mode requires that a word length of 32 bits be used (regardless of the actual data type). Consequently, the fixed I2S frame rate of 64 bits per frame (word length (WL) can be any value) and DC of 1 are assumed.

61.8.5 Receive Interrupt Enable Bit Description

When the RIE and RE bit are set, the processor is interrupted when either of the SSI Receive FIFO Full (RFF0/1) bits in SSI.SISR is set (if the corresponding Receive FIFO is enabled).

If the Receive FIFO is not enabled, the interrupt is generated when the corresponding SSI Receive Data Ready (RDR0/1) bit in the SSI.SISR is set. When the receive FIFO is enabled, a maximum of 15 values are available to be read (15 values per channel in Two-Channel mode). If not enabled, then one value can be read from the SRX register (one each in case of Two-Channel mode). If the RIE bit is cleared, these interrupts are disabled. However, the RFF0/1 and RDR0/1 bits still indicate the receive data register full condition. Reading the SSI.SRX registers clears the RDR bits, thus clearing the pending interrupt. Two receive data interrupts (two per channel in case of Two-Channel mode) are available: receive data with exception status and receive data without exception. The tables below show the conditions under which these interrupts are generated.

Table 61-8. SSI Receive Data 1 Interrupts

Interrupt	RIE	ROE0	RFF0/RDR0
Receive Data 1(with Exception Status)	1	1	1
Receive Data 1(without exception)	1	0	1

Table 61-9. SSI Receive Data 0 Interrupts

Interrupt	RIE	ROE1	RFF1/RDR1
Receive Data 0 (with Exception Status)	1	1	1
Receive Data 0 (without exception)	1	0	1

61.8.6 Transmit Interrupt Enable Bit Description

The SSI Transmit Interrupt Enable (TIE) control bit determines whether the processor is interrupted when the SSI transmitter needs to be serviced.

When the TIE and TE bits are set, the program controller is interrupted when either of the SSI Transmit FIFO Empty (TFE0/1) flags in SISR are set (if corresponding Transmit FIFO is enabled). If the corresponding Transmit FIFO is not enabled, an interrupt is generated when the corresponding SSI Transmit Data Register Empty (TDE0/1) flag in the SISR is set and Transmit Enable (TE) bit is set.

When Transmit FIFO 0 is enabled, a maximum of 15 values can be written to the SSI (15 per channel in case of Two-Channel mode, using Tx FIFO 1). If not enabled, then one value can be written to the SSI.STX0 register (one per channel in case of Two-Channel mode using SSI.STX1). When the TIE bit is cleared, all transmit interrupts are disabled. However, the TDE0/1 bits always indicate the corresponding SSI.STX register empty condition, even when the transmitter is disabled by the Transmit Enable (TE) bit (in the SSI.SCR). Writing data to the STX clears the corresponding TDE bit, thus clearing the interrupt. Two transmit data interrupts are available (four in case of Two-Channel mode, two per channel): transmit data with exception status and transmit data without exceptions. The tables below show the conditions under which these interrupts are generated.

Table 61-10. SSI Transmit Data 1 Interrupts

Interrupt	TIE	TUE1	TFE1/TDE1
Transmit Data 1 (with Exception Status)	1	1	1
Transmit Data 1 (without exception)	1	0	1

Table 61-11. SSI Transmit Data 0 Interrupts

Interrupt	TIE	TUE0	TFE0/TDE0
Transmit Data 0 (with Exception Status)	1	1	1
Transmit Data 0 (without exception)	1	0	1

61.8.7 Internal Frame and Clock Shutdown

During transmit/receive operation, disabling TE/RE will ensure that data transmission/ reception stops after current frame ends following which TFRC/RFRC Status bits will get set to indicate the Frame Completion State.

If TE is disabled 4 clock cycles before the next frame, extra frame generated are invalid frames. TFR_CLK_DIS/RFR_CLK_DIS bit is set in the current or any of the previous frames, SSI will stop driving the STFS/SRFS and STCK/SRCK signals after the current frame ends.

If TFR_CLK_DIS/RFR_CLK_DIS bit is not set, SSI will continue generating STFS/ SRFS and STCK/SRCK signals (in case direction is from SSI), which then can be disabled by writing '1' to TFR_CLK_DIS/RFR_CLK_DIS bit. SSI will then stop driving these signals after end of frame is reached following which TFRC/RFRC status bits will get set to indicate the Frame Completion State.

The following figure is an illustration of transmission case where TXDIR and TFDIR are both set to '1'. In this case TE is disabled with TFR_CLK_DIS bit set in current or any of the previous frames.

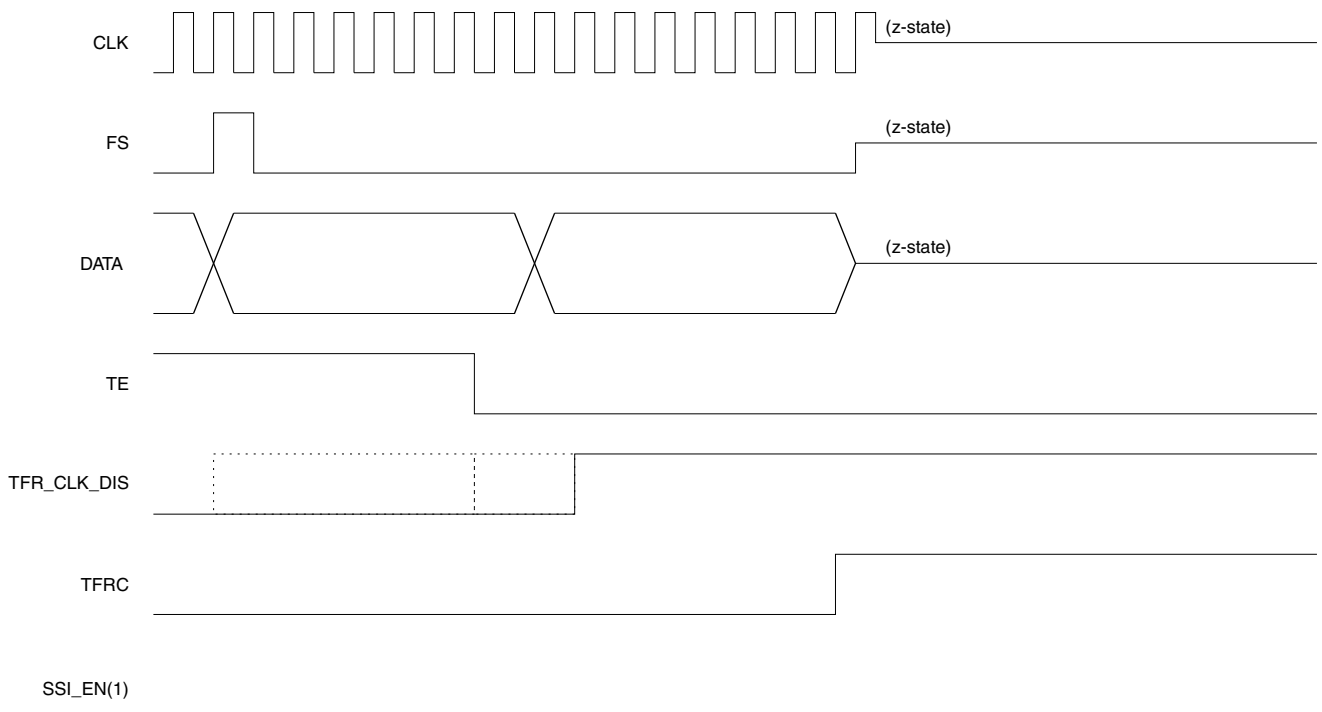


Figure 61-25. TFR_CLK_DIS assertion in current or previous frame as TE disable

The figure below is an illustration of transmission case where TXDIR and TFDIR are both set to '1'. In this case TFR_CLK_DIS bit is set after few frames of disabling TE. TFRC (Transmit Frame Complete) is set at frame boundary after TE is cleared. Once software services this interrupt and sets TFR_CLK_DIS bit later, TFRC bit is again set at next frame boundary.

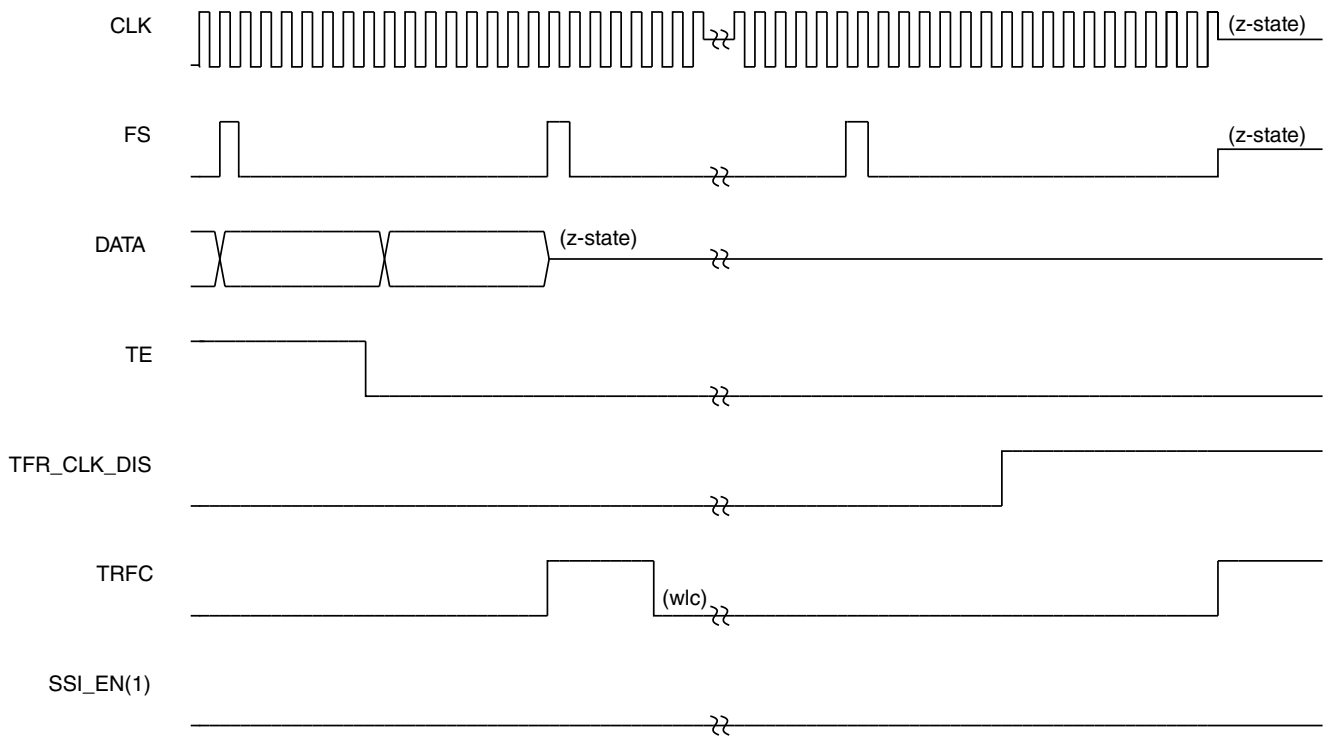


Figure 61-26. TFR_CLK_DIS assertion in subsequent frame after disabling TE

61.8.8 Peripheral Bus Interface

The SSI has a Peripheral Bus interface to provide a control and data interface. This interface is used by both the processor and DMA controller.

61.8.8.1 Transfer Lengths Supported

The Peripheral Bus interface of the SSI only supports 32-bit transfers with all SSI registers other than SSI.STX0, SSI.STX1, SSI.SRX0, and SSI.SRX1 (that is, the data registers).

With the exception of the data registers, using 8-bit and 16-bit transactions could result in undesired behavior but will not result in a transfer bus error. The data registers (SSI.STX0, SSI.STX1, SSI.SRX0, and SSI.SRX1) support 8-bit, 16-bit, and 32-bit transfer lengths without restrictions.

61.8.8.2 Transfer Bus Errors

Transfer bus errors are generated upon response to the following:

- Write transfer to a read-only register.
- Read or write access to a register space beyond the last populated register of the SSI in its memory map (up until the end of the allocated memory address range of the SSI).

61.8.8.3 Clock Rate

The Peripheral Bus clock frequency must be at least five times the serial bit clock frequency.

61.8.9 Reset

The SSI is affected by the following types of reset:

- Power-on Reset-The Power-on reset clears the SSIEN bit in SSI.SCR, which disables the SSI. All other status and control bits in the SSI are affected as described in SSI Programming Model in the "Memory Map and Register Definition section".
- SSI Reset-The SSI reset is generated when the SSIEN bit in the SSI.SCR is cleared. The SSI status bits are preset to the same state produced by the Power-on reset. The SSI control bits are unaffected. The control bits in the SSI.SCR are also unaffected. The SSI reset is useful for selective reset of the SSI without changing the present SSI control bits and without affecting the other peripherals.

61.9 SSI Memory Map/Register Definition

SSI memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
202_8000	SSI Transmit Data Register n (SSI1_STX0)	32	R/W	0000_0000h	61.9.1/5130
202_8004	SSI Transmit Data Register n (SSI1_STX1)	32	R/W	0000_0000h	61.9.1/5130
202_8008	SSI Receive Data Register n (SSI1_SRX0)	32	R	0000_0000h	61.9.2/5130
202_800C	SSI Receive Data Register n (SSI1_SRX1)	32	R	0000_0000h	61.9.2/5130
202_8010	SSI Control Register (SSI1_SCR)	32	R/W	0000_0000h	61.9.3/5131
202_8014	SSI Interrupt Status Register (SSI1_SISR)	32	w1c	0000_3003h	61.9.4/5133
202_8018	SSI Interrupt Enable Register (SSI1_SIER)	32	R/W	0000_3003h	61.9.5/5139
202_801C	SSI Transmit Configuration Register (SSI1_STCR)	32	R/W	0000_0200h	61.9.6/5143
202_8020	SSI Receive Configuration Register (SSI1_SRCR)	32	R/W	0000_0200h	61.9.7/5145

Table continues on the next page...

SSI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
202_8024	SSI Transmit Clock Control Register (SSI1_STCCR)	32	R/W	0004_0000h	61.9.8/5147
202_8028	SSI Receive Clock Control Register (SSI1_SRCCR)	32	R/W	0004_0000h	61.9.9/5149
202_802C	SSI FIFO Control/Status Register (SSI1_SFCSR)	32	R/W	0081_0081h	61.9.10/ 5150
202_8038	SSI AC97 Control Register (SSI1_SACNT)	32	R/W	0000_0000h	61.9.11/ 5154
202_803C	SSI AC97 Command Address Register (SSI1_SACADD)	32	R/W	0000_0000h	61.9.12/ 5155
202_8040	SSI AC97 Command Data Register (SSI1_SACDAT)	32	R/W	0000_0000h	61.9.13/ 5155
202_8044	SSI AC97 Tag Register (SSI1_SATAG)	32	R/W	0000_0000h	61.9.14/ 5156
202_8048	SSI Transmit Time Slot Mask Register (SSI1_STMSK)	32	R/W	0000_0000h	61.9.15/ 5156
202_804C	SSI Receive Time Slot Mask Register (SSI1_SRMSK)	32	R/W	0000_0000h	61.9.16/ 5157
202_8050	SSI AC97 Channel Status Register (SSI1_SACCST)	32	R	0000_0000h	61.9.17/ 5157
202_8054	SSI AC97 Channel Enable Register (SSI1_SACCEN)	32	W	0000_0000h	61.9.18/ 5158
202_8058	SSI AC97 Channel Disable Register (SSI1_SACCDIS)	32	W	0000_0000h	61.9.19/ 5158
202_C000	SSI Transmit Data Register n (SSI2_STX0)	32	R/W	0000_0000h	61.9.1/5130
202_C004	SSI Transmit Data Register n (SSI2_STX1)	32	R/W	0000_0000h	61.9.1/5130
202_C008	SSI Receive Data Register n (SSI2_SRX0)	32	R	0000_0000h	61.9.2/5130
202_C00C	SSI Receive Data Register n (SSI2_SRX1)	32	R	0000_0000h	61.9.2/5130
202_C010	SSI Control Register (SSI2_SCR)	32	R/W	0000_0000h	61.9.3/5131
202_C014	SSI Interrupt Status Register (SSI2_SISR)	32	w1c	0000_3003h	61.9.4/5133
202_C018	SSI Interrupt Enable Register (SSI2_SIER)	32	R/W	0000_3003h	61.9.5/5139
202_C01C	SSI Transmit Configuration Register (SSI2_STCR)	32	R/W	0000_0200h	61.9.6/5143
202_C020	SSI Receive Configuration Register (SSI2_SRCR)	32	R/W	0000_0200h	61.9.7/5145
202_C024	SSI Transmit Clock Control Register (SSI2_STCCR)	32	R/W	0004_0000h	61.9.8/5147
202_C028	SSI Receive Clock Control Register (SSI2_SRCCR)	32	R/W	0004_0000h	61.9.9/5149
202_C02C	SSI FIFO Control/Status Register (SSI2_SFCSR)	32	R/W	0081_0081h	61.9.10/ 5150
202_C038	SSI AC97 Control Register (SSI2_SACNT)	32	R/W	0000_0000h	61.9.11/ 5154
202_C03C	SSI AC97 Command Address Register (SSI2_SACADD)	32	R/W	0000_0000h	61.9.12/ 5155
202_C040	SSI AC97 Command Data Register (SSI2_SACDAT)	32	R/W	0000_0000h	61.9.13/ 5155

Table continues on the next page...

SSI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
202_C044	SSI AC97 Tag Register (SSI2_SATAG)	32	R/W	0000_0000h	61.9.14/ 5156
202_C048	SSI Transmit Time Slot Mask Register (SSI2_STMSK)	32	R/W	0000_0000h	61.9.15/ 5156
202_C04C	SSI Receive Time Slot Mask Register (SSI2_SRMSK)	32	R/W	0000_0000h	61.9.16/ 5157
202_C050	SSI AC97 Channel Status Register (SSI2_SACCST)	32	R	0000_0000h	61.9.17/ 5157
202_C054	SSI AC97 Channel Enable Register (SSI2_SACCEN)	32	W	0000_0000h	61.9.18/ 5158
202_C058	SSI AC97 Channel Disable Register (SSI2_SACCDIS)	32	W	0000_0000h	61.9.19/ 5158
203_0000	SSI Transmit Data Register n (SSI3_STX0)	32	R/W	0000_0000h	61.9.1/5130
203_0004	SSI Transmit Data Register n (SSI3_STX1)	32	R/W	0000_0000h	61.9.1/5130
203_0008	SSI Receive Data Register n (SSI3_SRX0)	32	R	0000_0000h	61.9.2/5130
203_000C	SSI Receive Data Register n (SSI3_SRX1)	32	R	0000_0000h	61.9.2/5130
203_0010	SSI Control Register (SSI3_SCR)	32	R/W	0000_0000h	61.9.3/5131
203_0014	SSI Interrupt Status Register (SSI3_SISR)	32	w1c	0000_3003h	61.9.4/5133
203_0018	SSI Interrupt Enable Register (SSI3_SIER)	32	R/W	0000_3003h	61.9.5/5139
203_001C	SSI Transmit Configuration Register (SSI3_STCR)	32	R/W	0000_0200h	61.9.6/5143
203_0020	SSI Receive Configuration Register (SSI3_SRCR)	32	R/W	0000_0200h	61.9.7/5145
203_0024	SSI Transmit Clock Control Register (SSI3_STCCR)	32	R/W	0004_0000h	61.9.8/5147
203_0028	SSI Receive Clock Control Register (SSI3_SRCCR)	32	R/W	0004_0000h	61.9.9/5149
203_002C	SSI FIFO Control/Status Register (SSI3_SFCSR)	32	R/W	0081_0081h	61.9.10/ 5150
203_0038	SSI AC97 Control Register (SSI3_SACNT)	32	R/W	0000_0000h	61.9.11/ 5154
203_003C	SSI AC97 Command Address Register (SSI3_SACADD)	32	R/W	0000_0000h	61.9.12/ 5155
203_0040	SSI AC97 Command Data Register (SSI3_SACDAT)	32	R/W	0000_0000h	61.9.13/ 5155
203_0044	SSI AC97 Tag Register (SSI3_SATAG)	32	R/W	0000_0000h	61.9.14/ 5156
203_0048	SSI Transmit Time Slot Mask Register (SSI3_STMSK)	32	R/W	0000_0000h	61.9.15/ 5156
203_004C	SSI Receive Time Slot Mask Register (SSI3_SRMSK)	32	R/W	0000_0000h	61.9.16/ 5157
203_0050	SSI AC97 Channel Status Register (SSI3_SACCST)	32	R	0000_0000h	61.9.17/ 5157
203_0054	SSI AC97 Channel Enable Register (SSI3_SACCEN)	32	W	0000_0000h	61.9.18/ 5158
203_0058	SSI AC97 Channel Disable Register (SSI3_SACCDIS)	32	W	0000_0000h	61.9.19/ 5158

61.9.1 SSI Transmit Data Register n (SSIx_STX*n*)

NOTE

Enable SSI (SSIEN=1) before writing to SSI Transmit Data Registers.

Address: Base address + 0h offset + (4d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	STXn																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SSIx_STX*n* field descriptions

Field	Description
STXn	SSI Transmit Data. These bits store the data to be transmitted by the SSI. These are implemented as the first word of their respective Tx FIFOs. Data written to these registers is transferred to the Transmit Shift Register (TXSR), when shifting of the previous data is complete. If both FIFOs are in use, data is alternately transferred from STX0 and STX1, to TXSR. Multiple writes to the STX registers will not result in the previous data being over-written by the subsequent data. STX1 can only be used in Two-Channel mode of operation. Protection from over-writing is present irrespective of whether the transmitter is enabled or not. Example 1: If Tx FIFO0 is in use and user writes Data1...Data16 to STX0,Data16 will not over-write Data1. Data1...Data15 are stored in the FIFO whileData16 is discarded. Example 2: If Tx FIFO0 is not in use and user writes Data1, Data2 to STX0, then Data2 will not over-write Data1 and will be discarded.

61.9.2 SSI Receive Data Register n (SSIx_SRX*n*)

Address: Base address + 8h offset + (4d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SRXn																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SSIx_SRX*n* field descriptions

Field	Description
SRXn	SSI Receive Data. These bits store the data received by the SSI. These are implemented as the first word of their respective Rx FIFOs. These bits receive data from the RXSR depending on the mode of operation. In case both FIFOs are in use, data is transferred to each data register alternately. SRX1 can only be used in Two-Channel mode of operation.

61.9.3 SSI Control Register (SSIx_SCR)

The SSI Control Register (SSI_SCR) sets up the SSI reset is controlled by bit 0 in the SSI_SCR. SSI operating modes are also selected in this register (except AC97 mode which is selected in the SSI_SACNT register).

Address: Base address + 10h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			SYNC_TX_FS	RFR_CLK_DIS	TFR_CLK_DIS	CLK_IST	TCH_EN	SYS_CLK_EN	I2S_MODE	SYN	NET	RE	TE	SSIEN	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SSIx_SCR field descriptions

Field	Description
31–13 Reserved	This read-only field is reserved and always has the value 0.
12 SYNC_TX_FS	SYNC_FS_TX bit provides a safe window for TE to be visible to the internal circuit which is just after FS occurrence. When SYNC_TX_FS is set, TE(SCR[1]) gets latched on FS occurrence & latched TE is used to enable/disable SSI transmitter. TE needs setup of 2 bit-clock cycles before occurrence of FS. If TE is changed within 2 bit-clock cycles of FS occurrence, there is high probability that TE will be latched on next FS. Note: With TFR_CLK_DIS feature on, TE is used directly to enable transmitter in following cases (i) Sync mode & Rx disabled (ii) Async Mode. Latched-TE is used to disable the transmitter. This bit has no relevance in gated mode and AC97 mode. 0 **TE_NOT_LATCHED** — TE not latched with FS occurrence & used directly for transmitter enable/ disable. 1 **TE_LATCHED** — TE latched with FS occurrence & latched-TE used for transmitter enable/disable.
11 RFR_CLK_DIS	Receive Frame Clock Disable. This bit provides the option to keep the Frame-sync and Clock enabled or to disable them after the receive frame in which the receiver is disabled. Writing to this bit has effect only when RE is disabled.The receiver is disabled by clearing the RE bit.

Table continues on the next page...

SSIx_SCR field descriptions (continued)

Field	Description
	0 **CONTINUE** — Continue Frame-sync/Clock generation after current frame during which RE is cleared. This may be required when Frame-sync and Clocks are required from SSI, even when no data is to be received. 1 **STOP** — Stop Frame-sync/Clock generation at next frame boundary. This will be effective also in case where receiver is already disabled in current or previous frames.
10 TFR_CLK_DIS	Transmit Frame Clock Disable. This bit provide option to keep the Frame-sync and Clock enabled or disabled after current transmit frame, in which transmitter is disabled by clearing TE bit. Writing to this bit has effect only when SSI is enabled TE is disabled. 0 **CONTINUE** — Continue Frame-sync/Clock generation after current frame during which TE is cleared. This may be required when Frame-sync and Clocks are required from SSI, even when no data is to be received. 1 **STOP** — Stop Frame-sync/Clock generation at next frame boundary. This will be effective also in case where transmitter is already disabled in current or previous frames.
9 CLK_IST	Clock Idle State. This bit controls the idle state of the transmit clock port during SSI internal gated mode. Note: When Clock idle state is '1' the clock polarity should always be negedge triggered and when Clock idle = '0' the clock polarity should always be positive edge triggered. 0 **IDLE_0** — Clock idle state is '0'. 1 **IDLE_1** — Clock idle state is '1'.
8 TCH_EN	Two-Channel Operation Enable. This bit allows SSI to operate in the two-channel mode.In this mode while receiving, the RXSR transfers data to SRX0 and SRX1 alternately and while transmitting, data is alternately transferred from STX0 and STX1 to TXSR. For an even number of slots, Two-Channel Operation can be enabled to optimize usage of both FIFOs or disabled as in the case of odd number of active slots. This feature is especially useful in I2S mode, where data for Left Speaker can be placed in Tx-FIFO0 and for Right speaker in Tx-FIFO1. 0 **DISABLED** — Two-channel mode disabled. 1 **ENABLED** — Two-channel mode enabled.
7 SYS_CLK_EN	Network Clock (Oversampling Clock) Enable. When set, this bit allows the SSI to output the network clock at the SRCK port, provided that synchronous mode, and transmit internal clock mode are set. The relationship between bit clock and network clock is determined by DIV2, PSR, and PM bits. This feature is especially useful in I2S Master mode to output network clock (oversampling clock) on SRCK port. 0 **NOT_OUTPUT** — network clock not output on SRCK port. 1 **OUTPUT** — network clock output on SRCK port.
6–5 I2S_MODE	I2S Mode Select. These bits allow the SSI to operate in Normal, I2S Master or I2S Slave mode. Refer to I2S Mode for a detailed description of I2S Mode of operation. Refer to Table 61-5 for details regarding settings.
4 SYN	Synchronous Mode. This bit controls whether SSI is in synchronous mode or not. In synchronous mode, the transmit and receive sections of SSI share a common clock port (STCK) and frame sync port (STFS). 0 **ASYNC_MODE** — Asynchronous mode selected. 1 **SYNC_MODE** — Synchronous mode selected.
3 NET	Network Mode. This bit controls whether SSI is in network mode or not. 0 **DISABLED** — Network mode not selected. 1 **ENABLED** — Network mode selected.

Table continues on the next page...

SSIx_SCR field descriptions (continued)

Field	Description
2 RE	Receive Enable. This control bit enables the receive section of the SSI. When this bit is enabled, data reception starts with the arrival of the next frame sync. If data is being received when this bit is cleared, data reception continues until the end of the current frame and then stops. If this bit is set again before the second to last bit of the last time slot in the current frame, then reception continues without interruption. RE should not be toggled in the same frame. 0 **DISABLED** — Receive section disabled. 1 **ENABLED** — Receive section enabled.
1 TE	Transmit Enable. This control bit enables the transmit section of the SSI. It enables the transfer of the contents of the STX registers to the TXSR and also enables the internal transmit clock. The transmit section is enabled when this bit is set and a frame boundary is detected. When this bit is cleared, the transmitter continues to send data until the end of the current frame and then stops. Data can be written to the STX registers with the TE bit cleared (the corresponding TDE bit will be cleared). If the TE bit is cleared and then set again before the second to last bit of the last time slot in the current frame, data transmission continues without interruption. The normal transmit enable sequence is to write data to the STX register(s) and then set the TE bit. The normal disable sequence is to clear the TE and TIE bits after the TDE bit is set. In gated clock mode, clearing the TE bit results in the clock stopping after the data currently in TXSR has shifted out. When the TE bit is set, the clock starts immediately (for internal gated clock mode). TE should not be toggled in the same frame. After enabling/disabling transmission, SSI expects 4 setup clock cycles before arrival of frame-sync for frame-sync to be accepted/rejected by In case of fewer clock cycles, there is high probability of the frame-sync to get missed. Note: If continuous clock is not provided, SSI expects 6 clock cycles before arrival of frame-sync for frame-sync to be accepted by the SSI. 0 **DISABLED** — Transmit section disabled. 1 **ENABLED** — Transmit section enabled.
0 SSIEN	SSIEN - SSI Enable This bit is used to enable/disable the SSI. When disabled, all SSI status bits are preset to the same state produced by the power-on reset, all control bits are unaffected, the contents of Tx and Rx FIFOs are cleared. When SSI is disabled, all internal clocks are disabled (except register access clock). 0 **DISABLED** — SSI is disabled. 1 **ENABLED** — SSI is enabled.

61.9.4 SSI Interrupt Status Register (SSIx_SISR)

The SSI Interrupt Status Register (SSI_SISR) is used to monitor the SSI. This register is used by the core to interrogate the status of the SSI. In gated mode of operation the TFS, RFS, TLS, RLS, TFRC and RFRC bits of AISR register are not generated. The status bits are described in the following table.

- SSI Status flags are valid when SSI is enabled.

- See Receive Interrupt Enable Bit Description and Transmit Interrupt Enable Bit Description for interrupt source mapping.
- All the flags in the SSI_SISR are updated after the first bit of the next SSI word has completed transmission or reception. Certain status bits (ROE0/1 and TUE0/1) are cleared by writing 1 to the corresponding interrupt status bit in SSI_SISR.

Address: Base address + 14h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0							RFRC	TFRC	0				CMDAU	CMDDU	RXT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RDR1	RDR0	TDE1	TDE0	ROE1	ROE0	TUE1	TUE0	TFS	RFS	TLS	RLS	RFF1	RFF0	TFE1	TFE0
W					w1c	w1c	w1c	w1c								
Reset	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1

SSIx_SISR field descriptions

Field	Description
31–25 Reserved	This read-only field is reserved and always has the value 0.
24 RFRC	Receive Frame Complete. This flag is set at the end of the frame during which Receiver is disabled. If Receive Frame & Clock are not disabled in the same frame, this flag is also set at the end of the frame in which Receive Frame & Clock are disabled. See the description of RFR_CLK_DIS bit for more details on how to disable Receiver Frame & Clock or keep them enabled after receiver is disabled.

Table continues on the next page...

SSIx_SISR field descriptions (continued)

Field	Description
	0 End of Frame not reached 1 End of frame reached after disabling RE or disabling RFR_CLK_DIS, when receiver is already disabled.
23 TFRC	Transmit Frame Complete. This flag is set at the end of the frame during which Transmitter is disabled. If Transmit Frame & Clock are not disabled in the same frame, this flag is also set at the end of the frame in which Transmit Frame & Clock are disabled. See description of TFR_CLK_DIS bit for more details on how to disable Transmit Frame & Clock or keep them enabled after transmitter is disabled. 0 End of Frame not reached 1 End of frame reached after disabling TE or disabling TFR_CLK_DIS, when transmitter is already disabled.
22–19 Reserved	This read-only field is reserved and always has the value 0.
18 CMDAU	Command Address Register Updated. This bit causes the Command Address Updated interrupt (when CMDAU_EN bit is set). This status bit is set each time there is a difference in the previous and current value of the received Command Address. This bit is cleared on reading the SACADD register. 0 No change in SACADD register. 1 SACADD register updated with different value.
17 CMDDU	Command Data Register Updated. This bit causes the Command Data Updated interrupt (when CMDDU_EN bit is set). This status bit is set each time there is a difference in the previous and current value of the received Command Data. This bit is cleared on reading the SACDAT register. 0 No change in SACDAT register. 1 SACDAT register updated with different value.
16 RXT	Receive Tag Updated. This status bit is set each time there is a difference in the previous and current value of the received tag. It causes the Receive Tag Interrupt (if RXT_EN bit is set). This bit is cleared on reading the SATAG register. 0 No change in SATAG register. 1 SATAG register updated with different value.
15 RDR1	Receive Data Ready 1. This flag bit is set when SRX1 or Rx FIFO 1 is loaded with a new value and Two-Channel mode is selected. RDR1 is cleared when the Core reads the SRX1 register. If Rx FIFO 1 is enabled, RDR1 is cleared when the FIFO is empty. If RIE and RDR1_EN are set, a Receive Data 1 interrupt request is issued on setting of RDR1 bit in case Rx FIFO1 is disabled, if the FIFO is enabled, the interrupt is issued on RFF1 assertion. The RDR1 bit is cleared by POR and SSI reset. 0 No new data for Core to read. 1 New data for Core to read.
14 RDR0	Receive Data Ready 0. This flag bit is set when SRX0 or Rx FIFO 0 is loaded with a new value. RDR0 is cleared when the Core reads the SRX0 register. If Rx FIFO 0 is enabled, RDR0 is cleared when the FIFO is empty. If RIE and RDR0_EN are set, a Receive Data 0 interrupt request is issued on setting of RDR0 bit in case Rx FIFO0 is disabled, if the FIFO is enabled, the interrupt is issued on RFF0 assertion. The RDR0 bit is cleared by POR and SSI reset. 0 No new data for Core to read. 1 New data for Core to read.

Table continues on the next page...

SSIx_SISR field descriptions (continued)

Field	Description
13 TDE1	Transmit Data Register Empty 1. This flag is set whenever data is transferred to TXSR from STX1 register and Two-Channel mode is selected. If Tx FIFO1 is enabled, this occurs when there is at least one empty slot in STX1 or Tx FIFO1. If Tx FIFO1 is not enabled, this occurs when the contents of STX1 are transferred to TXSR. The TDE1 bit is cleared when the Core writes to STX1. If TIE and TDE1_EN are set, an SSI Transmit Data 1 interrupt request is issued on setting of TDE1 bit. The TDE1 bit is cleared by POR and SSI reset. 0 Data available for transmission. 1 Data needs to be written by the Core for transmission.
12 TDE0	Transmit Data Register Empty 0. This flag is set whenever data is transferred to TXSR from STX0 register. If Tx FIFO 0 is enabled, this occurs when there is at least one empty slot in STX0 or Tx FIFO 0. If Tx FIFO 0 is not enabled, this occurs when the contents of STX0 are transferred to TXSR. The TDE0 bit is cleared when the Core writes to STX0. If TIE and TDE0_EN are set, an SSI Transmit Data 0 interrupt request is issued on setting of TDE0 bit. The TDE0 bit is cleared by POR and SSI reset. 0 Data available for transmission. 1 Data needs to be written by the Core for transmission.
11 ROE1	Receiver Overrun Error 1. This flag is set when the RXSR is filled and ready to transfer to SRX1 register or to Rx FIFO 1 (when enabled) and these are already full and Two-Channel mode is selected. If Rx FIFO 1 is enabled, this is indicated by RFF1 flag, else this is indicated by the RDR1 flag. The RXSR is not transferred in this case. The ROE1 flag causes an interrupt if RIE and ROE1_EN are set. The ROE1 bit is cleared by POR and SSI reset. It is also cleared by writing '1' to this bit. Clearing the RE bit does not affect the ROE1 bit. 0 Default interrupt issued to the Core. 1 Exception interrupt issued to the Core.
10 ROE0	Receiver Overrun Error 0. This flag is set when the RXSR is filled and ready to transfer to SRX0 register or to Rx FIFO 0 (when enabled) and these are already full. If Rx FIFO 0 is enabled, this is indicated by RFF0 flag, else this is indicated by the RDR0 flag. The RXSR is not transferred in this case. The ROE0 flag causes an interrupt if RIE and ROE0_EN are set. The ROE0 bit is cleared by POR and SSI reset. It is also cleared by writing '1' to this bit. Clearing the RE bit does not affect the ROE0 bit. 0 Default interrupt issued to the Core. 1 Exception interrupt issued to the Core.
9 TUE1	Transmitter Underrun Error 1. This flag is set when the TXSR is empty (no data to be transmitted), the TDE1 flag is set, a transmit time slot occurs and the SSI is in Two-Channel mode. When a transmit underrun error occurs, the previous data is retransmitted. In Network mode, each time slot requires data transmission (unless masked through STMSK register), when the transmitter is enabled (TE is set). The TUE1 flag causes an interrupt if TIE and TUE1_EN are set. The TUE1 bit is cleared by POR and SSI reset. It is also cleared by writing '1' to this bit. 0 Default interrupt issued to the Core. 1 Exception interrupt issued to the Core.
8 TUE0	Transmitter Underrun Error 0. This flag is set when the TXSR is empty (no data to be transmitted), the TDE0 flag is set and a transmit time slot occurs. When a transmit underrun error occurs, the previous data

Table continues on the next page...

SSIx_SISR field descriptions (continued)

Field	Description
	is retransmitted. In Network mode, each time slot requires data transmission (unless masked through STMSK register), when the transmitter is enabled (TE is set). The TUE0 flag causes an interrupt if TIE and TUE0_EN are set. The TUE0 bit is cleared by POR and SSI reset. It is also cleared by writing '1' to this bit. 0 Default interrupt issued to the Core. 1 Exception interrupt issued to the Core.
7 TFS	Transmit Frame Sync. This flag indicates the occurrence of transmit frame sync. Data written to the STX registers during the time slot when the TFS flag is set, is sent during the second time slot (in Network mode) or in the next first time slot (in Normal mode). In Network mode, the TFS bit is set during transmission of the first time slot of the frame and is then cleared when starting transmission of the next time slot. In Normal mode, this bit is high for the first time slot. This flag causes an interrupt if TIE and TFS_EN are set. The TFS bit is cleared by POR and SSI reset. 0 No Occurrence of Transmit frame sync. 1 Transmit frame sync occurred during transmission of last word written to STX registers.
6 RFS	Receive Frame Sync. This flag indicates the occurrence of receive frame sync. In Network mode, the RFS bit is set when the first slot of the frame is being received. It is cleared when the next slot begins to be received. In Normal mode, this bit is always high (When DC = 0). This flag causes an interrupt if RIE and RFS_EN are set. The RFS bit is cleared by POR and SSI reset. 0 No Occurrence of Receive frame sync. 1 Receive frame sync occurred during reception of next word in SRX registers.
5 TLS	Transmit Last Time Slot. This flag indicates the last time slot in a frame. When set, it indicates that the current time slot is the last time slot of the frame. TLS is set at the start of the last transmit time slot and causes the SSI to issue an interrupt (if TIE and TLS_EN are set). TLS is not generated when frame rate is 1 in normal mode of operation. TLS is cleared when the SISR is read with this bit set. The TLS bit is cleared by POR and SSI reset. 0 Current time slot is not last time slot of frame. 1 Current time slot is the last transmit time slot of frame.
4 RLS	Receive Last Time Slot. This flag indicates the last time slot in a frame. When set, it indicates that the current time slot is the last receive time slot of the frame. RLS is set at the end of the last time slot and causes the SSI to issue an interrupt (if RIE and RLS_EN are set). RLS is cleared when the SISR is read with this bit set. The RLS bit is cleared by POR and SSI reset. 0 Current time slot is not last time slot of frame. 1 Current time slot is the last receive time slot of frame.
3 RFF1	Receive FIFO Full 1. This flag is set when Rx FIFO1 is enabled, the data level in Rx FIFO1 reaches the selected Rx FIFO WaterMark 1 (RFWM1) threshold and the SSI is in Two-Channel mode. The setting of RFF1 only causes an interrupt when RIE and RFF1_EN are set, Rx FIFO1 is enabled and the Two-Channel mode is selected. RFF1 is automatically cleared when the amount of data in Rx FIFO1 falls below the threshold. The RFF1 bit is cleared by POR and SSI reset. When Rx FIFO1 contains15 words, the maximum it can hold, all further data received (for storage in this FIFO) is ignored until the FIFO contents are read. 0 **NOT_FULL** — Space available in Receive FIFO1. 1 **FULL** — Receive FIFO1 is full.
2 RFF0	Receive FIFO Full 0. This flag is set when Rx FIFO0 is enabled and the data level in Rx FIFO0 reaches the selected Rx FIFO WaterMark 0 (RFWM0) threshold. The setting of RFF0 only causes an interrupt

Table continues on the next page...

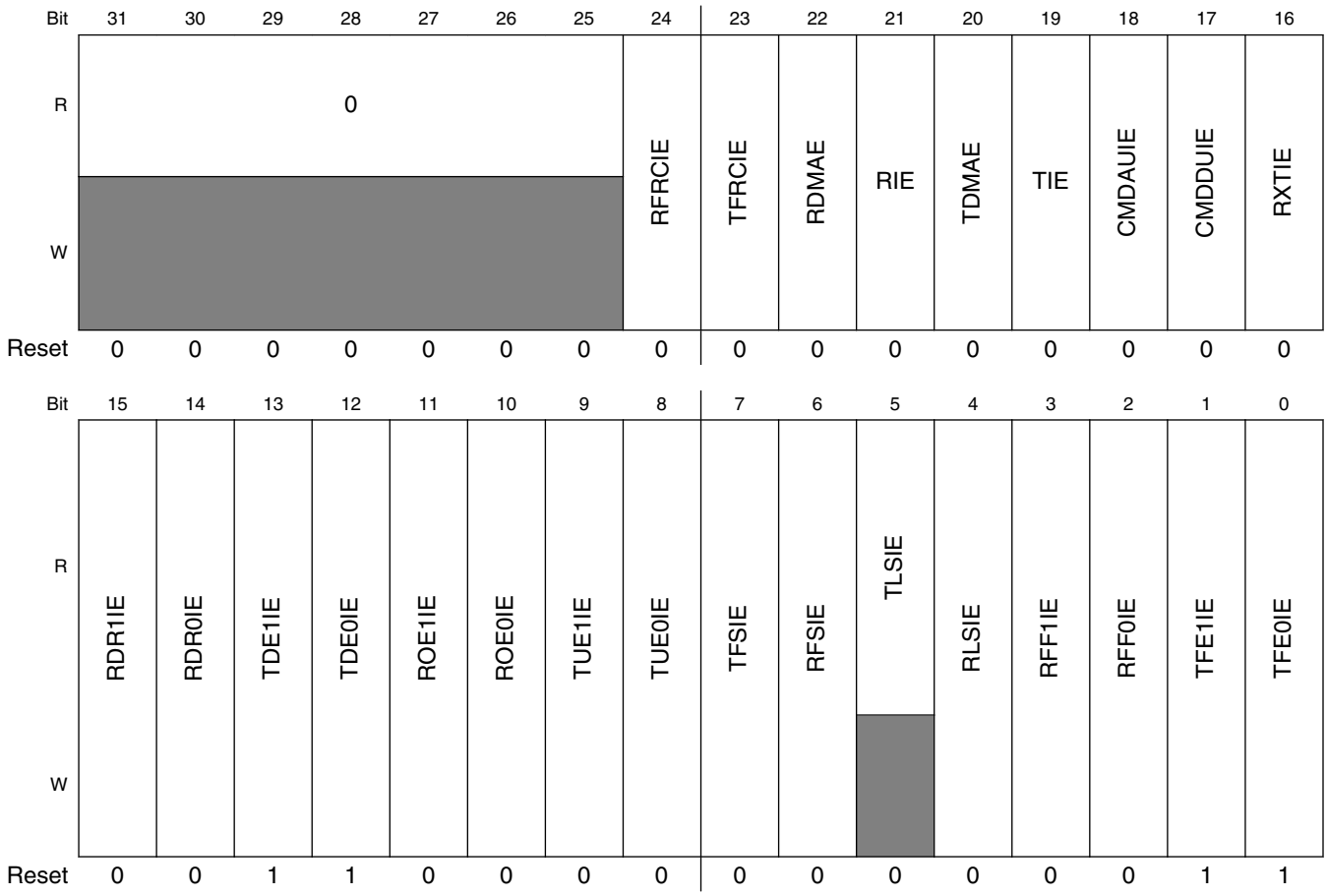
SSIx_SISR field descriptions (continued)

Field	Description
	when RIE and RFF0_EN are set and Rx FIFO0 is enabled. RFF0 is automatically cleared when the amount of data in Rx FIFO0 falls below the threshold. The RFF0 bit is cleared by POR and SSI reset. When Rx FIFO0 contains15 words, the maximum it can hold, all further data received (for storage in this FIFO) is ignored until the FIFO contents are read. 0 **NOT_FULL** — Space available in Receive FIFO0. 1 **FULL** — Receive FIFO0 is full.
1 TFE1	Transmit FIFO Empty 1. This flag is set when the empty slots in Tx FIFO exceed or are equal to the selected Tx FIFO WaterMark 1 (TFWM1) threshold and the Two-Channel mode is selected. The setting of TFE1 only causes an interrupt when TIE and TFE1_EN are set, Tx FIFO1 is enabled and Two-Channel mode is selected. The TFE1 bit is automatically cleared when the data level in Tx FIFO1 becomes more than the amount specified by the watermark bits. The TFE1 bit is set by POR and SSI reset. 0 **HAS_DATA** — Transmit FIFO1 has data for transmission. 1 **EMPTY** — Transmit FIFO1 is empty.
0 TFE0	Transmit FIFO Empty 0. This flag is set when the empty slots in Tx FIFO exceed or are equal to the selected Tx FIFO WaterMark 0 (TFWM0) threshold. The setting of TFE0 only causes an interrupt when TIE and TFE0_EN are set and Tx FIFO0 is enabled. The TFE0 bit is automatically cleared when the data level in Tx FIFO0 becomes more than the amount specified by the watermark bits. The TFE0 bit is set by POR and SSI reset. 0 **HAS_DATA** — Transmit FIFO0 has data for transmission. 1 **EMPTY** — Transmit FIFO0 is empty.

61.9.5 SSI Interrupt Enable Register (SSIx_SIER)

The SSI Interrupt Enable Register (SIER) is a 25-bit register used to set up the SSI interrupts and DMA requests.

Address: Base address + 18h offset



SSIx_SIER field descriptions

Field	Description
31–25 Reserved	This read-only field is reserved and always has the value 0.
24 RFRCIE	Receive Frame Complete Interrupt Enable. Enable Bit. Controls whether the corresponding status bit in SISR can issue an interrupt to the core or not. 0 Corresponding status bit cannot issue interrupt. 1 Corresponding status bit can issue interrupt.
23 TFRCIE	Transmit Frame Complete Interrupt Enable. Enable Bit. Controls whether the corresponding status bit in SISR can issue an interrupt to the core or not.

Table continues on the next page...

SSIx_SIER field descriptions (continued)

Field	Description
	0 Corresponding status bit cannot issue interrupt. 1 Corresponding status bit can issue interrupt.
22 RDMAE	Receive DMA Enable. This bit allows SSI to request for DMA transfers. When enabled, DMA requests are generated when any of the RFF0/1 bits in the SISR are set and if the corresponding RFEN bit is also set. If the corresponding FIFO is disabled, a DMA request is generated when the corresponding RDR bit is set. 0 SSI Receiver DMA requests disabled. 1 SSI Receiver DMA requests enabled.
21 RIE	Receive Interrupt Enable. This control bit allows the SSI to issue receiver related interrupts to the Core. Refer to Receive Interrupt Enable Bit Description for a detailed description of this bit. 0 SSI Receiver Interrupt requests disabled. 1 SSI Receiver Interrupt requests enabled.
20 TDMAE	Transmit DMA Enable. This bit allows SSI to request for DMA transfers. When enabled, DMA requests are generated when any of the TFE0/1 bits in the SISR are set and if the corresponding TFEN bit is also set. If the corresponding FIFO is disabled, a DMA request is generated when the corresponding TDE bit is set. 0 SSI Transmitter DMA requests disabled. 1 SSI Transmitter DMA requests enabled.
19 TIE	Transmit Interrupt Enable. This control bit allows the SSI to issue transmitter data related interrupts to the Core. Refer to Transmit Interrupt Enable Bit Description for a detailed description of this bit. 0 SSI Transmitter Interrupt requests disabled. 1 SSI Transmitter Interrupt requests enabled.
18 CMDAUIE	Command Address Register Updated Interrupt Enable. Enable Bit. Controls whether the corresponding status bit in SISR can issue an interrupt to the core or not. 0 Corresponding status bit cannot issue interrupt. 1 Corresponding status bit can issue interrupt.
17 CMDDUIE	Command Data Register Updated Interrupt Enable. Enable Bit. Controls whether the corresponding status bit in SISR can issue an interrupt to the core or not. 0 Corresponding status bit cannot issue interrupt. 1 Corresponding status bit can issue interrupt.
16 RXTIE	Receive Tag Updated Interrupt Enable. Enable Bit. Controls whether the corresponding status bit in SISR can issue an interrupt to the core or not. 0 Corresponding status bit cannot issue interrupt. 1 Corresponding status bit can issue interrupt.
15 RDR1IE	Receive Data Ready 1 Interrupt Enable. Enable Bit. Controls whether the corresponding status bit in SISR can issue an interrupt to the core or not. 0 Corresponding status bit cannot issue interrupt. 1 Corresponding status bit can issue interrupt.
14 RDR0IE	Receive Data Ready 0 Interrupt Enable. Enable Bit. Controls whether the corresponding status bit in SISR can issue an interrupt to the core or not.

Table continues on the next page...

SSIx_SIER field descriptions (continued)

Field	Description
	0 Corresponding status bit cannot issue interrupt. 1 Corresponding status bit can issue interrupt.
13 TDE1IE	Transmit Data Register Empty 1 Interrupt Enable. Enable Bit. Controls whether the corresponding status bit in SISR can issue an interrupt to the core or not. 0 Corresponding status bit cannot issue interrupt. 1 Corresponding status bit can issue interrupt.
12 TDE0IE	Transmit Data Register Empty 0 Interrupt Enable. Enable Bit. Controls whether the corresponding status bit in SISR can issue an interrupt to the core or not. 0 Corresponding status bit cannot issue interrupt. 1 Corresponding status bit can issue interrupt.
11 ROE1IE	Receiver Overrun Error 1 Interrupt Enable. Enable Bit. Controls whether the corresponding status bit in SISR can issue an interrupt to the core or not. 0 Corresponding status bit cannot issue interrupt. 1 Corresponding status bit can issue interrupt.
10 ROE0IE	Receiver Overrun Error 0 Interrupt Enable. Enable Bit. Controls whether the corresponding status bit in SISR can issue an interrupt to the core or not. 0 Corresponding status bit cannot issue interrupt. 1 Corresponding status bit can issue interrupt.
9 TUE1IE	Transmitter Underrun Error 1 Interrupt Enable. Enable Bit. Controls whether the corresponding status bit in SISR can issue an interrupt to the core or not. 0 Corresponding status bit cannot issue interrupt. 1 Corresponding status bit can issue interrupt.
8 TUE0IE	Transmitter Underrun Error 0 Interrupt Enable. Enable Bit. Controls whether the corresponding status bit in SISR can issue an interrupt to the core or not. 0 Corresponding status bit cannot issue interrupt. 1 Corresponding status bit can issue interrupt.
7 TFSIE	Transmit Frame Sync Interrupt Enable. Enable Bit. Controls whether the corresponding status bit in SISR can issue an interrupt to the core or not. 0 Corresponding status bit cannot issue interrupt. 1 Corresponding status bit can issue interrupt.
6 RFSIE	Receive Frame Sync Interrupt Enable. Enable Bit. Controls whether the corresponding status bit in SISR can issue an interrupt to the core or not. 0 Corresponding status bit cannot issue interrupt. 1 Corresponding status bit can issue interrupt.
5 TLSIE	Transmit Last Time Slot Interrupt Enable. Enable Bit. Controls whether the corresponding status bit in SISR can issue an interrupt to the core or not.

Table continues on the next page...

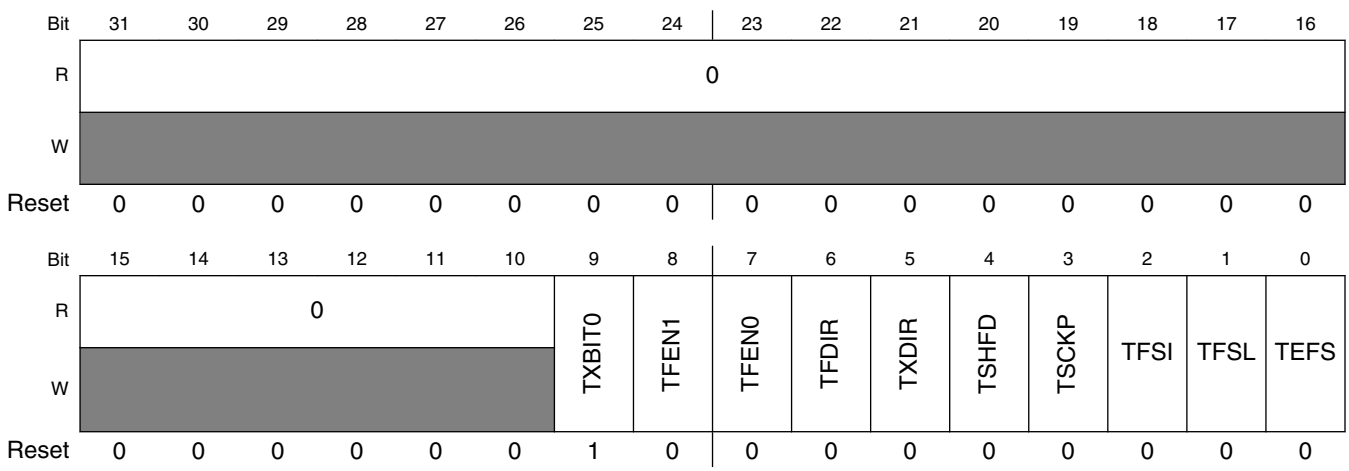
SSIx_SIER field descriptions (continued)

Field	Description
	0 Corresponding status bit cannot issue interrupt. 1 Corresponding status bit can issue interrupt.
4 RLSIE	Receive Last Time Slot Interrupt Enable. Enable Bit. Controls whether the corresponding status bit in SISR can issue an interrupt to the core or not. 0 Corresponding status bit cannot issue interrupt. 1 Corresponding status bit can issue interrupt.
3 RFF1IE	Receive FIFO Full 1 Interrupt Enable. Enable Bit. Controls whether the corresponding status bit in SISR can issue an interrupt to the core or not. 0 Corresponding status bit cannot issue interrupt. 1 Corresponding status bit can issue interrupt.
2 RFF0IE	Receive FIFO Full 0 Interrupt Enable. Enable Bit. Controls whether the corresponding status bit in SISR can issue an interrupt to the core or not. 0 Corresponding status bit cannot issue interrupt. 1 Corresponding status bit can issue interrupt.
1 TFE1IE	Transmit FIFO Empty 1 Interrupt Enable. Enable Bit. Controls whether the corresponding status bit in SISR can issue an interrupt to the core or not. 0 Corresponding status bit cannot issue interrupt. 1 Corresponding status bit can issue interrupt.
0 TFE0IE	Transmit FIFO Empty 0 Interrupt Enable. Enable Bit. Controls whether the corresponding status bit in SISR can issue an interrupt to the core or not. 0 Corresponding status bit cannot issue interrupt. 1 Corresponding status bit can issue interrupt.

61.9.6 SSI Transmit Configuration Register (SSIx_STCR)

The SSI Transmit Configuration Register (SSI_STCR) is a read/write control register used to direct the transmit operation of the STCR controls the direction of the bit clock and frame sync ports, STCK and STFS. Interrupt enable bit for the transmit sections is provided in this control register. The Power-on reset clears all SSI_STCR bits. However, SSI reset does not affect the SSI_STCR bits. The SSI_STCR bits are described in the following paragraphs. See the Programmable Registers section for the programming model of the SSI. The SSI Control Register (SSI_SCR) must first be set to enable interrupts. Next, the SSI interrupt bit in the Interrupt Enable Register (SSI_SIER) must be set to enable the interrupt. Finally, the interrupt can be enabled from within the SSI.

Address: Base address + 1Ch offset



SSIx_STCR field descriptions

Field	Description
31–10 Reserved	This read-only field is reserved and always has the value 0.
9 TXBIT0	Transmit Bit 0. This control bit allows SSI to transmit the data word from bit position 0 or 15/31 in the transmit shift register. The shifting data direction can be MSB or LSB first, controlled by the TSHFD bit. 0 **MSB_ALIGNED** — Shifting with respect to bit 31 (if word length = 16, 18, 20, 22 or 24) or bit 15 (if word length = 8, 10 or 12) of transmit shift register (MSB aligned). 1 **LSB_ALIGNED** — Shifting with respect to bit 0 of transmit shift register (LSB aligned).
8 TFEN1	Transmit FIFO Enable 1. This bit enables transmit FIFO 1. When enabled, the FIFO allows15 samples to be transmitted by the SSI (per channel) (a 9th sample can be shifting out) before TDE1 bit is set. When the FIFO is disabled, an interrupt is generated when a single sample is transferred to the transmit shift register (provided the interrupt is enabled). 0 Transmit FIFO 1 disabled. 1 Transmit FIFO 1 enabled.

Table continues on the next page...

SSIx_STCR field descriptions (continued)

Field	Description
7 TFEN0	Transmit FIFO Enable 0. This bit enables transmit FIFO 0. When enabled, the FIFO allows15 samples to be transmitted by the SSI per channel (a 9th sample can be shifting out) before TDE0 bit is set. When the FIFO is disabled, an interrupt is generated when a single sample is transferred to the transmit shift register (provided the interrupt is enabled). 0 Transmit FIFO 0 disabled. 1 Transmit FIFO 0 enabled.
6 TFDIR	Transmit Frame Direction. This bit controls the direction and source of the transmit frame sync signal. Internally generated frame sync signal is sent out through the STFS port and external frame sync is taken from the same port. 0 **EXTERNAL** — Frame Sync is external. 1 **INTERNAL** — Frame Sync generated internally.
5 TXDIR	Transmit Clock Direction. This bit controls the direction and source of the clock signal used to clock the TXSR. Internally generated clock is output through the STCK port. External clock is taken from this port. Refer to Table 61-2 for details of clock pin configurations. 0 **EXTERNAL** — Transmit Clock is external. 1 **INTERNAL** — Transmit Clock generated internally.
4 TSHFD	Transmit Shift Direction. This bit controls whether the MSB or LSB will be transmitted first in a sample. **NOTE:** The CODEC device labels the MSB as bit 0, whereas the Core labels the LSB as bit 0. Therefore, when using a standard CODEC, Core MSB (CODEC LSB) is shifted in first (TSHFD cleared). 0 **MSB_FIRST** — Data transmitted MSB first. 1 **LSB_FIRST** — Data transmitted LSB first.
3 TSCKP	Transmit Clock Polarity. This bit controls which bit clock edge is used to clock out data for the transmit section. Note: TSCKP is 0 CLK_IST = 0; TSCKP is 1CLK_IST = 1 0 **RISING_EDGE** — Data clocked out on rising edge of bit clock. 1 **FALLING_EDGE** — Data clocked out on falling edge of bit clock.
2 TFSI	Transmit Frame Sync Invert. This bit controls the active state of the frame sync I/O signal for the transmit section of SSI. 0 **ACTIVE_HIGH** — Transmit frame sync is active high. 1 **ACTIVE_LOW** — Transmit frame sync is active low.
1 TFSL	Transmit Frame Sync Length. This bit controls the length of the frame sync signal to be generated or recognized for the transmit section. The length of a word-long frame sync is same as the length of the data word selected by WL[3:0]. 0 **ONE_WORD** — Transmit frame sync is one-word long. 1 **ONE_CLOCK_BIT** — Transmit frame sync is one-clock-bit long.
0 TEFS	Transmit Early Frame Sync. This bit controls when the frame sync is initiated for the transmit section. The frame sync signal is deasserted after one bit-for-bit length frame sync and after one word-for-word length frame sync. In case of synchronous operation, the frame sync can also be initiated on receiving the first bit of data. 0 **FIRST_BIT** — Transmit frame sync initiated as the first bit of data is transmitted. 1 **ONE_BIT_BEFORE** — Transmit frame sync is initiated one bit before the data is transmitted.

61.9.7 SSI Receive Configuration Register (SSIx_SRCR)

The SSI Receive Configuration Register (SSI_SRCR) is a read/write control register used to direct the receive operation of the SSI. SSI.SRCR controls the direction of the bit clock and frame sync ports, SRCK and SRFS. Interrupt enable bit for the transmit sections is provided in this control register. The Power-on reset clears all SSI_SRCR bits. However, SSI reset does not affect the SSI_SRCR bits.

Address: Base address + 20h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0					RXEXT	RXBIT0	RFEN1	RFEN0	RFDIR	RXDIR	RSHFD	RSCKP	RFSI	RFSL	REFS
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

SSIx_SRCR field descriptions

Field	Description
31–11 Reserved	This read-only field is reserved and always has the value 0.
10 RXEXT	Receive Data Extension. This control bit allows SSI to store the received data word in sign extended form. This bit affects data storage only in case received data is LSB aligned (SRCR[9]=1) 0 **OFF** — Sign extension turned off. 1 **ON** — Sign extension turned on.
9 RXBIT0	Receive Bit 0. This control bit allows SSI to receive the data word at bit position 0 or 15/31 in the receive shift register. The shifting data direction can be MSB or LSB first, controlled by the RSHFD bit. 0 **MSB_ALIGNED** — Shifting with respect to bit 31 (if word length = 16, 18, 20, 22 or 24) or bit 15 (if word length = 8, 10 or 12) of receive shift register (MSB aligned). 1 **LSB_ALIGNED** — Shifting with respect to bit 0 of receive shift register (LSB aligned).
8 RFEN1	Receive FIFO Enable 1. This bit enables receive FIFO 1. When enabled, the FIFO allows15 samples to be received by the SSI per channel (a 16th sample can be shifting in) before RDR1 bit is set. When the FIFO is disabled, an interrupt is generated when a single sample is received by the SSI (provided the interrupt is enabled). 0 Receive FIFO 1 disabled. 1 Receive FIFO 1 enabled.

Table continues on the next page...

SSIx_SRCR field descriptions (continued)

Field	Description
7 RFEN0	Receive FIFO Enable 0. This bit enables receive FIFO 0. When enabled, the FIFO allows15 samples to be received by the SSI (per channel) (a 16th sample can be shifting in) before RDR0 bit is set. When the FIFO is disabled, an interrupt is generated when a single sample is received by the SSI (provided the interrupt is enabled). 0 Receive FIFO 0 disabled. 1 Receive FIFO 0 enabled.
6 RFDIR	Receive Frame Direction. This bit controls the direction and source of the receive frame sync signal. Internally generated frame sync signal is sent out through the SRFS port and external frame sync is taken from the same port. 0 **EXTERNAL** — Frame Sync is external. 1 **INTERNAL** — Frame Sync generated internally.
5 RXDIR	Receive Clock Direction. This bit controls the direction and source of the clock signal used to clock the RXSR. Internally generated clock is output through the SRCK port. External clock is taken from this port. Refer to Table 61-2 for details on clock pin configurations. 0 **EXTERNAL** — Receive Clock is external. 1 **INTERNAL** — Receive Clock generated internally.
4 RSHFD	Receive Shift Direction. This bit controls whether the MSB or LSB will be received first in a sample. **NOTE:** The CODEC device labels the MSB as bit 0, whereas the Core labels the LSB as bit 0. Therefore, when using a standard CODEC, Core MSB (CODEC LSB) is shifted in first (RSHFD cleared). 0 **MSB_FIRST** — Data received MSB first. 1 **LSB_FIRST** — Data received LSB first.
3 RSCKP	Receive Clock Polarity. This bit controls which bit clock edge is used to latch in data for the receive section. 0 **FALLING_EDGE** — Data latched on falling edge of bit clock. 1 **RISING_EDGE** — Data latched on rising edge of bit clock.
2 RFSI	Receive Frame Sync Invert. This bit controls the active state of the frame sync I/O signal for the receive section of SSI. 0 **ACTIVE_HIGH** — Receive frame sync is active high. 1 **ACTIVE_LOW** — Receive frame sync is active low.
1 RFSL	Receive Frame Sync Length. This bit controls the length of the frame sync signal to be generated or recognized for the receive section. The length of a word-long frame sync is same as the length of the data word selected by WL[3:0]. 0 **ONE_WORD** — Receive frame sync is one-word long. 1 **ONE_CLOCK_BIT** — Receive frame sync is one-clock-bit long.
0 REFS	Receive Early Frame Sync. This bit controls when the frame sync is initiated for the receive section. The frame sync is disabled after one bit-for-bit length frame sync and after one word-for-word length frame sync. 0 **FIRST_BIT** — Receive frame sync initiated as the first bit of data is received. 1 **ONE_BIT_BEFORE** — Receive frame sync is initiated one bit before the data is received.

61.9.8 SSI Transmit Clock Control Register (SSIx_STCCR)

The SSI Transmit and Receive Control (SSI_STCCR and SSI_SRCCR) registers are 19-bit, read/write control registers used to direct the operation of the SSI. The Clock Controller Module (CCM) can source the SSI clock (SSI's sys clock from CCM's ssi_clk_root) from multiple sources and perform fractional division to support commonly used audio bit rates. The CCM can maintain the SSI's sys clock frequency at a constant rate even in cases where the ipg_clk (from CCM) frequency changes. These registers control the SSI clock generator, bit and frame sync rates, word length, and number of words per frame for the serial data. The SSI.STCCR register is dedicated to the transmit section, and the SSI_SRCCR register is dedicated to the receive section except in Synchronous mode, in which the SSI_STCCR register controls both the receive and transmit sections. Power-on reset clears all SSI_STCCR and SSI_SRCCR bits. SSI reset does not affect the SSI_STCCR and SSI_SRCCR bits. The control bits are described in the following paragraphs. Although the bit patterns of the SSI_STCCR and SSI_SRCCR registers are the same, the contents of these two registers can be programmed differently.

Table 61-52. SSI Data Length

WL3	WL2	WL1	WL0	Number of Bits/Word	Supported in Implementation
0	0	0	0	2	No
0	0	0	1	4	No
0	0	1	0	6	No
0	0	1	1	8	Yes
0	1	0	0	10	Yes
0	1	0	1	12	Yes
0	1	1	0	14	No
0	1	1	1	16	Yes
1	0	0	0	18	Yes
1	0	0	1	20	Yes
1	0	1	0	22	Yes
1	0	1	1	24	Yes
1	1	0	0	26	No
1	1	0	1	28	No
1	1	1	0	30	No
1	1	1	1	32	No

Address: Base address + 24h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0													DIV2	PSR	WL3_ WL0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WL3_WL0			DC4_DC0					PM7_PM0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SSIx_STCCR field descriptions

Field	Description
31–19 Reserved	This read-only field is reserved and always has the value 0.
18 DIV2	Divide By 2. This bit controls a divide-by-two divider in series with the rest of the prescalers. 0 Divider bypassed. 1 Divider used to divide clock by 2.
17 PSR	Prescaler Range. This bit controls a fixed divide-by-eight prescaler in series with the variable prescaler. It extends the range of the prescaler for those cases where a slower bit clock is required. 0 Prescaler bypassed. 1 Prescaler used to divide clock by 8.
16–13 WL3_WL0	Word Length Control. These bits are used to control the length of the data words being transferred by the SSI. These bits control the Word Length Divider in the Clock Generator. They also control the frame sync pulse length when the FSL bit is cleared. In I2S Master mode, the SSI works with a fixed word length of 32, and the WL bits are used to control the amount of valid data in those 32 bits. In AC97 Mode of operation, if word length is set to any value other than 16 bits, it will result in a word length of 20 bits.
12–8 DC4_DC0	Frame Rate Divider Control. These bits are used to control the divide ratio for the programmable frame rate dividers. The divide ratio works on the word clock. In Normal mode, this ratio determines the word transfer rate. In Network mode, this ratio sets the number of words per frame. The divide ratio ranges from 1 to 32 in Normal mode and from 2 to 32 in Network mode. In Normal mode, a divide ratio of 1 (DC=00000) provides continuous periodic data word transfer. A bit-length frame sync must be used in this case. These bits can be programmed with values ranging from "00000" to "11111" to control the number of words in a frame.
PM7_PM0	Prescaler Modulus Select. These bits control the prescale divider in the clock generator. This prescaler is used only in Internal Clock mode to divide the internal clock. The bit clock output is available at the clock port. A divide ratio from 1 to 256 (PM[7:0] = 0x00 to 0xFF) can be selected. Refer to DIV2, PSR and PM Bit Description for details regarding settings.

61.9.9 SSI Receive Clock Control Register (SSIx_SRCCR)

The SSI Transmit and Receive Control (SSI_STCCR and SSI_SRCCR) registers are 19-bit, read/write control registers used to direct the operation of the SSI. The Clock Controller Module (CCM) can source the SSI clock (SSI's sys clock-from CCM's ssi_clk_root) from multiple sources and perform fractional division to support commonly used audio bit rates. The CCM can maintain the SSI's sys clock frequency at a constant rate even in cases where the ipg_clk from CCM frequency changes. These registers control the SSI clock generator, bit and frame sync rates, word length, and number of words per frame for the serial data. The SSI_STCCR register is dedicated to the transmit section, and the SSI_SRCCR register is dedicated to the receive section except in Synchronous mode, in which the SSI_STCCR register controls both the receive and transmit sections. Power-on reset clears all SSI_STCCR and SSI_SRCCR bits. SSI reset does not affect the SSI_STCCR and SSI_SRCCR bits. The control bits are described in the following paragraphs. Although the bit patterns of the SSI_STCCR and SSI_SRCCR registers are the same, the contents of these two registers can be programmed differently.

Table 61-54. SSI Data Length

WL3	WL2	WL1	WL0	Number of Bits/Word	Supported in Implementation
0	0	0	0	2	No
0	0	0	1	4	No
0	0	1	0	6	No
0	0	1	1	8	Yes
0	1	0	0	10	Yes
0	1	0	1	12	Yes
0	1	1	0	14	No
0	1	1	1	16	Yes
1	0	0	0	18	Yes
1	0	0	1	20	Yes
1	0	1	0	22	Yes
1	0	1	1	24	Yes
1	1	0	0	26	No
1	1	0	1	28	No
1	1	1	0	30	No
1	1	1	1	32	No

Address: Base address + 28h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0													DIV2	PSR	WL3_ WL0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WL3_WL0			DC4_DC0					PM7_PM0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SSIx_SRCCR field descriptions

Field	Description
31–19 Reserved	This read-only field is reserved and always has the value 0.
18 DIV2	Divide By 2. This bit controls a divide-by-two divider in series with the rest of the prescalers. 0 Divider bypassed. 1 Divider used to divide clock by 2.
17 PSR	Prescaler Range. This bit controls a fixed divide-by-eight prescaler in series with the variable prescaler. It extends the range of the prescaler for those cases where a slower bit clock is required. 0 Prescaler bypassed. 1 Prescaler used to divide clock by 8.
16–13 WL3_WL0	Word Length Control. These bits are used to control the length of the data words being transferred by the SSI. These bits control the Word Length Divider in the Clock Generator. They also control the frame sync pulse length when the FSL bit is cleared. In I2S Master mode, the SSI works with a fixed word length of 32, and the WL bits are used to control the amount of valid data in those 32 bits. In AC97 Mode of operation, if word length is set to any value other than 16 bits, it will result in a word length of 20 bits.
12–8 DC4_DC0	Frame Rate Divider Control. These bits are used to control the divide ratio for the programmable frame rate dividers. The divide ratio works on the word clock. In Normal mode, this ratio determines the word transfer rate. In Network mode, this ratio sets the number of words per frame. The divide ratio ranges from 1 to 32 in Normal mode and from 2 to 32 in Network mode. In Normal mode, a divide ratio of 1 (DC=00000) provides continuous periodic data word transfer. A bit-length frame sync must be used in this case. These bits can be programmed with values ranging from "00000" to "11111" to control the number of words in a frame.
PM7_PM0	Prescaler Modulus Select. These bits control the prescale divider in the clock generator. This prescaler is used only in Internal Clock mode to divide the internal clock. The bit clock output is available at the clock port. A divide ratio from 1 to 256 (PM[7:0] = 0x00 to 0xFF) can be selected. Refer to DIV2, PSR and PM Bit Description for details regarding settings.

61.9.10 SSI FIFO Control/Status Register (SSIx_SFCSR)

The SSI FIFO Control / Status Register indicates the status of the Transmit FIFO Empty flag, with different settings of the Transmit FIFO WaterMark bits and varying amounts of data in the Tx FIFO

.

Table 61-56. Status of Transmit FIFO Empty Flag

Transmit FIFO Watermark (TFWM)	Number of data in Tx-Fifo														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
2	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
3	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
4	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
5	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
6	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
7	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
8	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
9	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
10	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
11	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
12	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
13	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
14	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Address: Base address + 2Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RFCNT1				TFCNT1				RFWM1				TFWM1				RFCNT0				TFCNT0				RFWM0				TFWM0			
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1

SSIx_SFCSR field descriptions

Field	Description
31–28 RFCNT1	Receive FIFO Counter1. These bits indicate the number of data words in Receive FIFO 1. 0000 0 data word in receive FIFO 0001 1 data word in receive FIFO 0010 2 data word in receive FIFO 0011 3 data word in receive FIFO 0100 4 data word in receive FIFO 0101 5 data word in receive FIFO 0110 6 data word in receive FIFO 0111 7 data word in receive FIFO 1000 8 data word in receive FIFO 1001 9 data word in receive FIFO 1010 10 data word in receive FIFO 1011 11 data word in receive FIFO 1100 12 data word in receive FIFO 1101 13 data word in receive FIFO

Table continues on the next page...

SSIx_SFCSR field descriptions (continued)

Field	Description
	1110 14 data word in receive FIFO 1111 15 data word in receive FIFO
27–24 TFCNT1	Transmit FIFO Counter1. These bits indicate the number of data words in Transmit FIFO. 0000 0 data word in transmit FIFO 0001 1 data word in transmit FIFO 0010 2 data word in transmit FIFO 0011 3 data word in transmit FIFO 0100 4 data word in transmit FIFO 0101 5 data word in transmit FIFO 0110 6 data word in transmit FIFO 0111 7 data word in transmit FIFO 1000 8 data word in transmit FIFO 1001 9 data word in transmit FIFO 1010 10 data word in transmit FIFO 1011 11 data word in transmit FIFO 1100 12 data word in transmit FIFO 1101 13 data word in transmit FIFO 1110 14 data word in transmit FIFO 1111 15 data word in transmit FIFO
23–20 RFWM1	Receive FIFO Full WaterMark 1. These bits control the threshold at which the RFF1 flag will be set. The RFF1 flag is set whenever the data level in Rx FIFO 1 reaches the selected threshold. 0000 Reserved 0001 RFF set when at least one data word has been written to the Receive FIFO. Set when RxFIFO = 1,2.....15 data words 0010 RFF set when 2 or more data words have been written to the Receive FIFO. Set when RxFIFO = 2,3.....15 data words 0011 RFF set when 3 or more data words have been written to the Receive FIFO. Set when RxFIFO = 3,4.....15 data words 0100 RFF set when 4 or more data words have been written to the Receive FIFO. Set when RxFIFO = 4,5.....15 data words 0101 RFF set when 5 or more data words have been written to the Receive FIFO. Set when RxFIFO = 5,6.....15 data words 0110 RFF set when 6 or more data words have been written to the Receive.. Set when RxFIFO = 6,7.....15 data words 0111 RFF set when 7 or more data words have been written to the Receive FIFO. Set when RxFIFO = 7,8.....15 data words 1000 RFF set when 8 or more data words have been written to the Receive FIFO. Set when RxFIFO =8,9.....15 data words 1001 RFF set when 9 or more data words have been written to the Receive FIFO. Set when RxFIFO = 9,10.....15 data words 1010 RFF set when 10 or more data words have been written to the Receive FIFO. Set when RxFIFO = 10,11.....15 data words 1011 RFF set when 11 or more data words have been written to the Receive FIFO. Set when RxFIFO = 11,12.....15 data words 1100 RFF set when 12 or more data words have been written to the Receive FIFO. Set when RxFIFO = 12,13.....15 data words

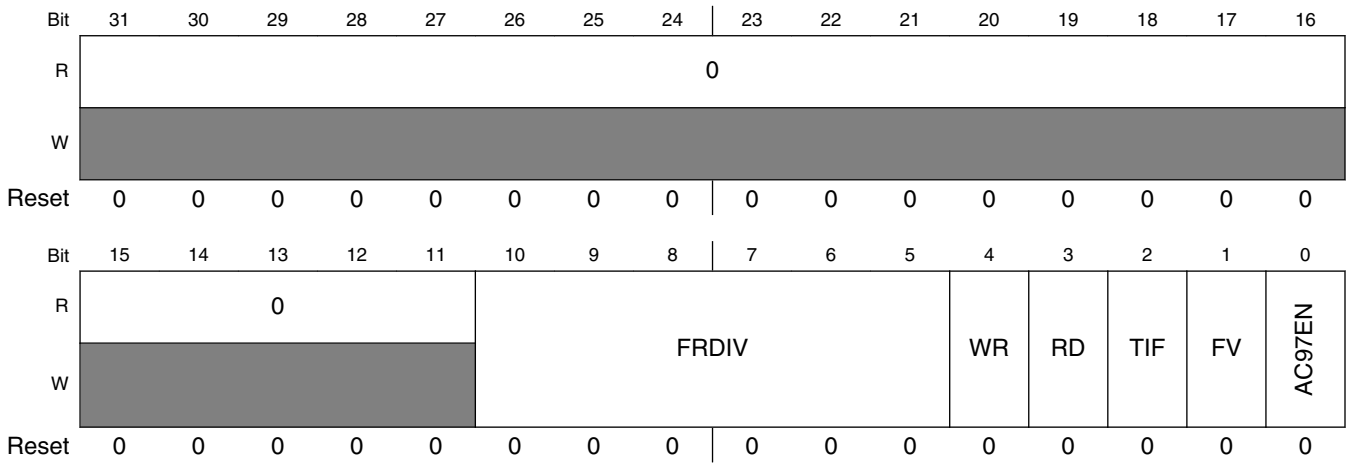
Table continues on the next page...

SSIx_SFCSR field descriptions (continued)

Field	Description
	1101 RFF set when 13 or more data words have been written to the Receive FIFO. Set when RxFIFO = 13,14,15data words 1110 RFF set when 14 or more data words have been written to the Receive FIFO. Set when RxFIFO = 14,15 data words 1111 RFF set when 15 data words have been written to the Receive FIFO (default). Set when RxFIFO = 15 data words
19–16 TFWM1	Transmit FIFO Empty WaterMark 1. These bits control the threshold at which the TFE1 flag will be set. The TFE1 flag is set whenever the empty slots in Tx FIFO exceed or are equal to the selected threshold. 0000 Reserved 0001 TFE set when there are more than or equal to 1 empty slots in Transmit FIFO (default). Transmit FIFO empty is set when TxFIFO <= 14 data. 0010 TFE set when there are more than or equal to 2 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <=13 data. 0011 TFE set when there are more than or equal to 3 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <=12 data. 0100 TFE set when there are more than or equal to 4 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <=11 data. 0101 TFE set when there are more than or equal to 5 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <=10 data. 0110 TFE set when there are more than or equal to 6 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <=9 data. 0111 TFE set when there are more than or equal to 7 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <=8 data. 1000 TFE set when there are more than or equal to 8 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <=7 data. 1001 TFE set when there are more than or equal to 9 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 6 data. 1010 TFE set when there are more than or equal to 10 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 5 data. 1011 TFE set when there are more than or equal to 11 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 4 data. 1100 TFE set when there are more than or equal to 12 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 3 data. 1101 TFE set when there are more than or equal to 13 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 2 data. 1110 TFE set when there are more than or equal to 14 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 1 data. 1111 TFE set when there are 15 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO = 0 data.
15–12 RFCNT0	Receive FIFO Counter 0. These bits indicate the number of data words in Receive FIFO 0. See SSI_SFCSR[RFCNT1] for details regarding settings for receive FIFO counter bits.
11–8 TFCNT0	Transmit FIFO Counter 0. These bits indicate the number of data words in Transmit FIFO 0. See SSI_SFCSR[TFCNT1] for details regarding settings for transmit FIFO counter bits.
7–4 RFWM0	Receive FIFO Full WaterMark 0. These bits control the threshold at which the RFF0 flag will be set. The RFF0 flag is set whenever the data level in Rx FIFO 0 reaches the selected threshold. See SSI_SFCSR[RFWM1] for details regarding settings for receive FIFO watermark bits.
TFWM0	Transmit FIFO Empty WaterMark 0. These bits control the threshold at which the TFE0 flag will be set. The TFE0 flag is set whenever the empty slots in Tx FIFO exceed or are equal to the selected threshold. See SSI_SFCSR[TFWM0] for details regarding settings for transmit FIFO watermark bits.

61.9.11 SSI AC97 Control Register (SSIx_SACNT)

Address: Base address + 38h offset



SSIx_SACNT field descriptions

Field	Description
31–11 Reserved	This read-only field is reserved and always has the value 0.
10–5 FRDIV	Frame Rate Divider. These bits control the frequency of AC97 data transmission/reception. They are programmed with the number of frames for which the SSI should be idle, after operating in one frame. Through these bits, AC97 frequency of operation, from 48 KHz (000000) to 1 KHz (101111) can be achieved. Sample Value: 001010 (10 Decimal) = SSI will operate once every 11 frames.
4 WR	Write Command. This bit specifies whether the next frame will carry an AC97 Write Command or not. The programmer should take care that only one of the bits (WR or RD) is set at a time. When this bit is set, the corresponding tag bits (corresponding to Command Address and Command Data slots of the next Tx frame) are automatically set. This bit is automatically cleared by the SSI after completing transmission of a frame. 0 Next frame will not have a Write Command. 1 Next frame will have a Write Command.
3 RD	Read Command. This bit specifies whether the next frame will carry an AC97 Read Command or not. The programmer should take care that only one of the bits (WR or RD) is set at a time. When this bit is set, the corresponding tag bit (corresponding to Command Address slot of the next Tx frame) is automatically set. This bit is automatically cleared by the SSI after completing transmission of a frame. 0 Next frame will not have a Read Command. 1 Next frame will have a Read Command.
2 TIF	Tag in FIFO. This bit controls the destination of the information received in AC97 tag slot (Slot #0). 0 **SATAG_REGISTER** — Tag info stored in SATAG register. 1 **RX_FIFO0** — Tag info stored in Rx FIFO 0.
1 FV	Fixed/Variable Operation. This bit selects whether the SSI is in AC97 Fixed mode or AC97 Variable mode.

Table continues on the next page...

SSIx_SACNT field descriptions (continued)

Field	Description
	0 **FIXED** — AC97 Fixed Mode. 1 **VARIABLE** — AC97 Variable Mode.
0 AC97EN	AC97 Mode Enable. This bit is used to enable SSI AC97 operation. Refer to AC97 Mode for details of AC97 operation. 0 AC97 mode disabled. 1 SSI in AC97 mode.

61.9.12 SSI AC97 Command Address Register (SSIx_SACADD)

Address: Base address + 3Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																SACADD															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SSIx_SACADD field descriptions

Field	Description
31–19 Reserved	This read-only field is reserved and always has the value 0.
SACADD	AC97 Command Address. These bits store the Command Address Slot information (bit 19 of the slot is sent in accordance with the Read and Write Command bits in SSI_SACNT register). These bits can be updated by a direct write from the Core. They are also updated with the information received in the incoming Command Address Slot. If the contents of these bits change due to an update, the CMDAU bit in SISR is set.

61.9.13 SSI AC97 Command Data Register (SSIx_SACDAT)

Address: Base address + 40h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																SACDAT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SSIx_SACDAT field descriptions

Field	Description
31–20 Reserved	This read-only field is reserved and always has the value 0.
SACDAT	AC97 Command Data. The outgoing Command Data Slot carries the information contained in these bits. These bits can be updated by a direct write from the Core. They are also updated with the information received in the incoming Command Data Slot. If the contents of these bits change due to an update, the

Table continues on the next page...

SSIx_SACDAT field descriptions (continued)

Field	Description
	CMDDU bit in SISR is set. These bits are transmitted only during AC97 Write Command. During AC97 Read Command, 0x00000 is transmitted in time slot #2.

61.9.14 SSI AC97 Tag Register (SSIx_SATAG)

Address: Base address + 44h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																SATAG															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SSIx_SATAG field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
SATAG	AC97 Tag Value. Writing to this register (by the Core) sets the value of the Tx-Tag in AC97 fixed mode of operation. On a read, the Core gets the Rx-Tag Value received (in the last frame) from the Codec. If TIF bit in SSI_SACNT register is set, the TAG value is also stored in Rx-FIFO in addition to SATAG register. When the received Tag value changes, the RXT bit in SISR register is set. Bits SATAG[1:0] convey the Codec -ID. In current implementation only Primary Codecs are supported. Thus writing value 2'b00 to this field is mandatory.

61.9.15 SSI Transmit Time Slot Mask Register (SSIx_STMSK)

Address: Base address + 48h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	STMSK																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SSIx_STMSK field descriptions

Field	Description
STMSK	Transmit Mask. These bits indicate which slot has been masked in the current frame. The Core can write to this register to control the time slots in which the SSI transmits data. Each bit has info corresponding to the respective time slot in the frame. Transmit mask bits should not be used in I2S Slave mode of operation. SSI_STMSK register value must be set before enabling Transmission. 0 Valid Time Slot. 1 Time Slot masked (no data transmitted in this time slot).

61.9.16 SSI Receive Time Slot Mask Register (SSIx_SRMSK)

Address: Base address + 4Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SRMSK																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SSIx_SRMSK field descriptions

Field	Description
SRMSK	Receive Mask. These bits indicate which slot has been masked in the current frame. The Core can write to this register to control the time slots in which the SSI receives data. Each bit has info corresponding to the respective time slot in the frame. SSI_SRMSK register value must be set before enabling Receiver. Receive mask bits should not be used in I2S Slave mode of operation. 0 Valid Time Slot. 1 Time Slot masked (no data received in this time slot).

61.9.17 SSI AC97 Channel Status Register (SSIx_SACCST)

Address: Base address + 50h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																						SACCST									
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SSIx_SACCST field descriptions

Field	Description
31–10 Reserved	This read-only field is reserved and always has the value 0.
SACCST	AC97 Channel Status. These bits indicate which data slot has been enabled in AC97 variable mode operation. This register is updated in case the core enables/disables a channel through a write to SSI_SACCEN/SSI_SACCDIS register or the external codec enables a channel by sending a '1' in the corresponding SLOTREQ bit. Bit [0] corresponds to the first data slot in an AC97 frame (Slot #3) and Bit [9] corresponds to the tenth data slot (slot #12). The contents of this register only have relevance while the SSI is operating in AC97 variable mode. Writes to this register result in an error response on the block interface. 0 Data channel disabled. 1 Data channel enabled.

61.9.18 SSI AC97 Channel Enable Register (SSIx_SACCEN)

Address: Base address + 54h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																						SACCEN									
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SSIx_SACCEN field descriptions

Field	Description
31–10 Reserved	This read-only field is reserved and always has the value 0.
SACCEN	AC97 Channel Enable. The Core writes a '1' to these bits to enable an AC97 data channel. Writing a '0' has no effect. Bit [0] corresponds to the first data slot in an AC97 frame (Slot #3) and Bit [9] corresponds to the tenth data slot (slot #12). Writes to these bits only have effect in the AC97 Variable mode of operation. These bits are always read as '0' by the Core. 0 Write Has no effect. 1 Write Enables the corresponding data channel.

61.9.19 SSI AC97 Channel Disable Register (SSIx_SACCDIS)

Address: Base address + 58h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																						SACCDIS									
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SSIx_SACCDIS field descriptions

Field	Description
31–10 Reserved	This read-only field is reserved and always has the value 0.
SACCDIS	AC97 Channel Disable. The Core writes a '1' to these bits to disable an AC97 data channel. Writing a '0' has no effect. Bit [0] corresponds to the first data slot in an AC97 frame (Slot #3) and Bit [9] corresponds to the tenth data slot (slot #12). Writes to these bits only have effect in the AC97 Variable mode of operation. These bits are always read as '0' by the Core. 0 Write Has no effect. 1 Write Disables the corresponding data channel.

Chapter 62
Temperature Monitor (TEMPMON)

62.1 Overview

The temperature sensor module implements a temperature sensor/conversion function based on a temperature-dependent voltage to time conversion.

The module features an alarm function that can raise an interrupt signal if the temperature is above a specified threshold. A self-repeating mode can also be programmed which executes a temperature sensing operation based on a programmed delay.

Software can use this module to monitor the on-die temperature and take appropriate actions such as throttling back the core frequency when a temperature interrupt is set.

The high-level implementation of the temperature sensor is shown in the figure below.

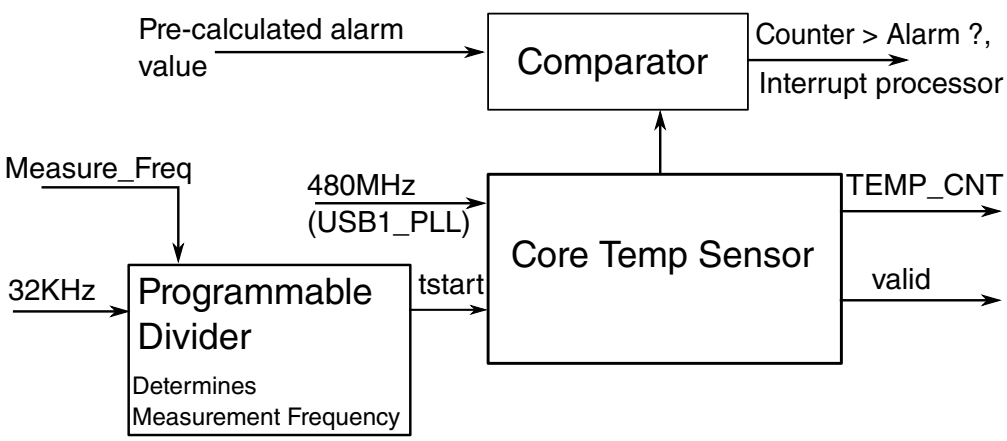


Figure 62-1. High Level Temp Sensor System Diagram

As shown in the figure above, the temperature sensor uses and assumes that the bandgap reference, 480MHz PLL and 32KHz RTC modules are properly programmed and fully settled for correct operation.

62.2 Software Usage Guidelines

During normal system operation software can use the temperature sensor counter output (TEMP_CNT) in conjunction with the fused temperature calibration data to determine the on-die operational temperature or to set an over-temperature interrupt alarm to within a couple of °C.

Based on calibration, two sets of temperature and counter values will be available via fuses on the device. These data points will correspond to the points (N_1, T_1) and (N_2, T_2) in the curve below.

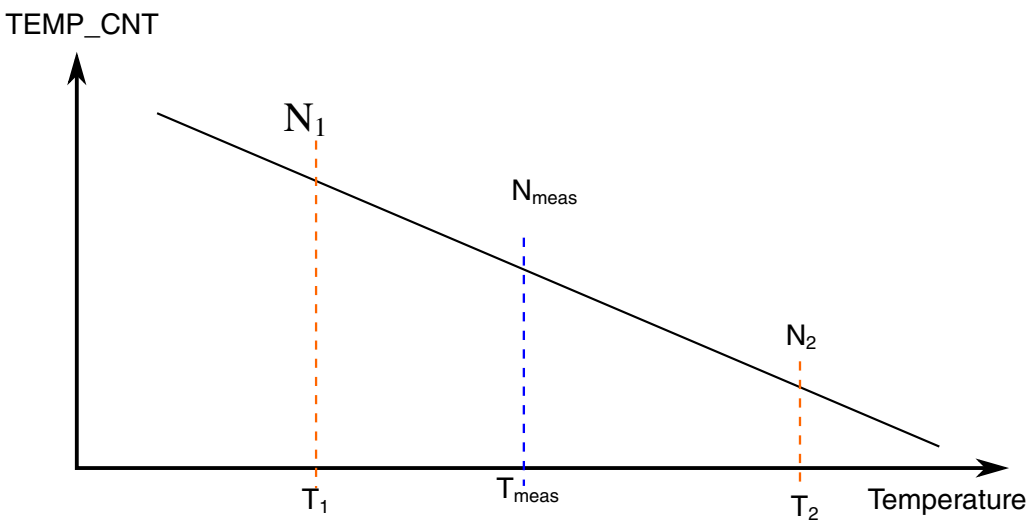


Figure 62-2. Temperature Measurement Cycle

After a temperature measurement cycle, software should use the calibration points in conjunction with the temperature code value in the TEMPMON_TEMPSENSE0[TEMP_CNT] bitfield to calculate the temperature for the device using the following equation:

$$T_{meas} = T_2 - (N_{meas} - N_2) * ((T_2 - T_1) / (N_1 - N_2))$$

Likewise, to determine the alarm counter value to be written in the TEMPMON_TEMPSENSE0 register for a temperature based interrupt, the above equation can be solved for the N_{meas} value that should be used based on the desired temperature trigger.

The temperature calibration point fuse values are available in the OCOTP_ANA1 register. The temperature calibration values are fused individually for each part in the product testing process. The fields of this register are described in the following table.

Table 62-1. OCOTP_ANA1 Temperature Sensor Calibration Data

Bit Range	Bit Mask	Name	Description
[31:20]	FFF0_0000h	ROOM_COUNT	Value of TEMPMON_TEMPSENSE0[TEMP_VALUE] after a measurement cycle at room temperature (25.0 °C).
[19:8]	000F_FF00h	HOT_COUNT	Value of TEMPMON_TEMPSENSE0[TEMP_VALUE] after a measurement cycle at the hot temperature, i.e. HOT_TEMP.
[7:0]	0000_00FFh	HOT_TEMP	The hot temperature test point. Each LSB equals 1 °C.

The points on the calibration curve are as follows.

- $(N_1, T_1) = (ROOM\_COUNT, 25.0)$
- $(N_2, T_2) = (HOT\_COUNT, HOT\_TEMP)$
- $(N_{meas}, T_{meas}) = (TEMP\_CNT, T_{meas})$

Substituting the fields from OCOTP_ANA1 into the earlier equation results in the following:

$$T_{meas} = HOT\_TEMP - (N_{meas} - HOT\_COUNT) * ((HOT\_TEMP - 25.0) / (ROOM\_COUNT – HOT\_COUNT))$$

62.3 TEMPMON Memory Map/Register Definition

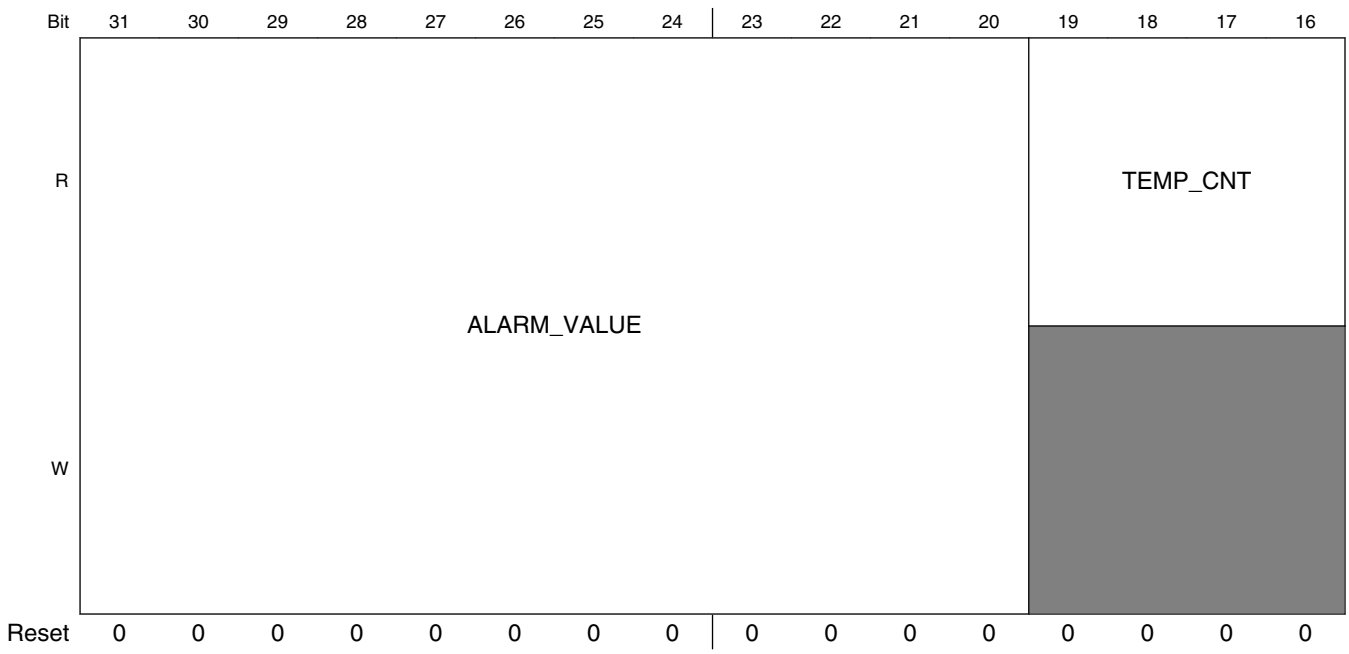
TEMPMON memory map

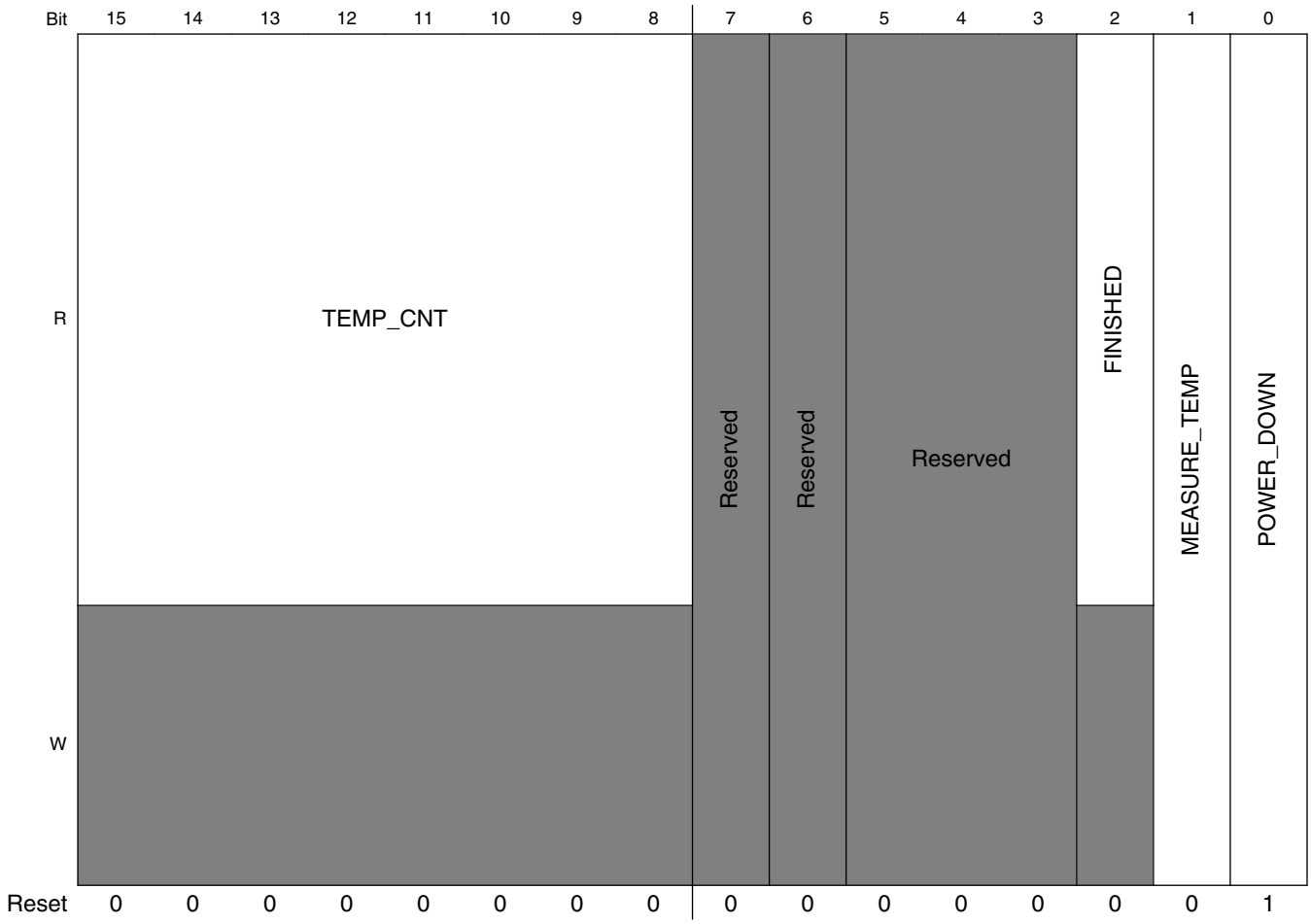
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20C_8180	Tempsensor Control Register 0 (TEMPMON_TEMPSENSE0)	32	R/W	0000_0001h	62.3.1/5162
20C_8184	Tempsensor Control Register 0 (TEMPMON_TEMPSENSE0_SET)	32	R/W	0000_0001h	62.3.1/5162
20C_8188	Tempsensor Control Register 0 (TEMPMON_TEMPSENSE0_CLR)	32	R/W	0000_0001h	62.3.1/5162
20C_818C	Tempsensor Control Register 0 (TEMPMON_TEMPSENSE0_TOG)	32	R/W	0000_0001h	62.3.1/5162
20C_8190	Tempsensor Control Register 1 (TEMPMON_TEMPSENSE1)	32	R/W	0000_0001h	62.3.2/5164
20C_8194	Tempsensor Control Register 1 (TEMPMON_TEMPSENSE1_SET)	32	R/W	0000_0001h	62.3.2/5164
20C_8198	Tempsensor Control Register 1 (TEMPMON_TEMPSENSE1_CLR)	32	R/W	0000_0001h	62.3.2/5164
20C_819C	Tempsensor Control Register 1 (TEMPMON_TEMPSENSE1_TOG)	32	R/W	0000_0001h	62.3.2/5164

62.3.1 Tempsensor Control Register 0 (TEMPMON_TEMPSENSE0*n*)

This register defines the basic controls for the temperature sensor minus the frequency of automatic sampling which is defined in the tempsensor.

Address: 20C_8000h base + 180h offset + (4d × i), where i=0d to 3d





TEMPMON_TEMPSENSE0*n* field descriptions

Field	Description
31–20 ALARM_VALUE	This bit field contains the temperature count (raw sensor output) that will generate an alarm interrupt.
19–8 TEMP_CNT	This bit field contains the last measured temperature count.
7 -	This field is reserved. Reserved.
6 -	This field is reserved. Reserved.
5–3 -	This field is reserved. Reserved
2 FINISHED	Indicates that the latest temp is valid. This bit should be cleared by the sensor after the start of each measurement. 0 **INVALID** — Last measurement is not ready yet. 1 **VALID** — Last measurement is valid.
1 MEASURE_ TEMP	Starts the measurement process. If the measurement frequency is zero in the TEMPSENSE1 register, this results in a single conversion.

Table continues on the next page...

TEMPMON_TEMPSENSE0*n* field descriptions (continued)

Field	Description
	0 **STOP** — Do not start the measurement process. 1 **START** — Start the measurement process.
0 POWER_DOWN	This bit powers down the temperature sensor. 0 **POWER_UP** — Enable power to the temperature sensor. 1 **POWER_DOWN** — Power down the temperature sensor.

62.3.2 Tempsensor Control Register 1 (TEMPMON_TEMPSENSE1*n*)

This register defines the automatic repeat time of the temperature sensor.

Address: 20C_8000h base + 190h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																MEASURE_FREQ															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

TEMPMON_TEMPSENSE1*n* field descriptions

Field	Description
31–16 -	This field is reserved. Reserved.
MEASURE_ FREQ	This bits determines how many RTC clocks to wait before automatically repeating a temperature measurement. The pause time before remeasuring is the field value multiplied by the RTC period. 0x0000 Defines a single measurement with no repeat. 0x0001 Updates the temperature value at a RTC clock rate. 0x0002 Updates the temperature value at a RTC/2 clock rate. . . . — 0xFFFF Determines a two second sample period with a 32.768KHz RTC clock. Exact timings depend on the accuracy of the RTC clock.

Chapter 63
TrustZone Address Space Controller (TZASC)

63.1 Overview

The TrustZone Address Space Controller (TZASC) protects security-sensitive SW and data in a trusted execution environment against potentially compromised SW running on the platform.

The TZASC block diagram is shown in figure below.

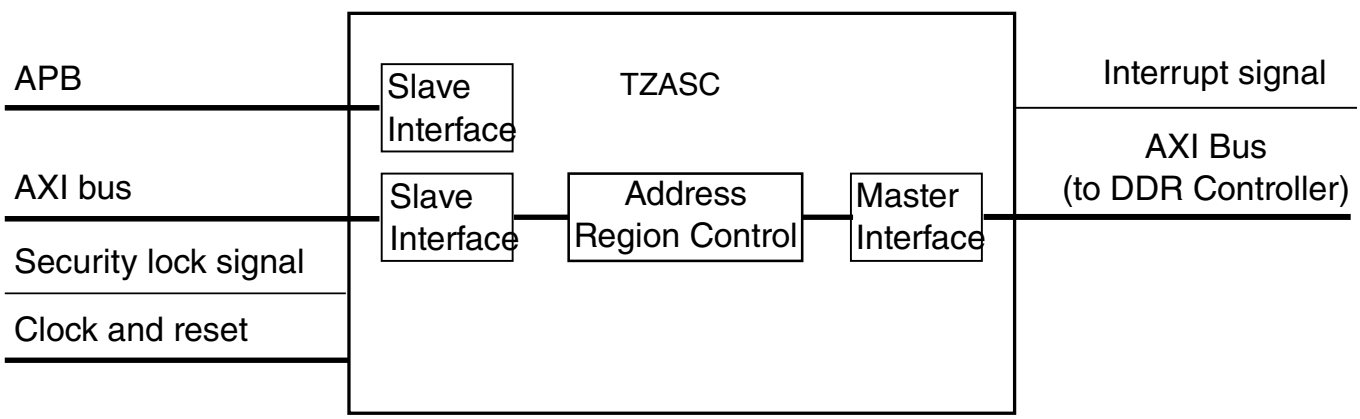


Figure 63-1. TZASC Block Diagram

The TZASC is an IP by ARM ("CoreLink™ TrustZone Address Space Controller TZC-380"), designed to provide configurable protection over program (SW) memory space.

The main features of TZASC are:

- Supports 16 independent address regions
- Access controls are independently programmable for each address region
- Sensitive registers may be locked
- Host interrupt may be programmed to signal attempted access control violations

- AXI master/slave interfaces for transactions
- APB slave interface for configuration and status reporting

63.2 Clocks

The table found here describes the clock sources for TZASC.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 63-1. TZASC Clocks

Clock name	Clock Root	Description
aclk	mmdc_ch0_axi_clk_root	Module clock

63.3 i.MX 6Dual/6Quad Specific Configuration

The i.MX 6Dual/6Quad uses two TZASC instances, one on each of the two DDR channels (MMDC0, MMDC1 modules), to provide address protection based on access security level.

The i.MX 6Dual/6Quad utilizes bus muxing logic to route DDR memory traffic through or bypass the TZASC modules. By default, the TZASC modules are bypassed, and their clocks are gated off.

TZASC1 provides protection on memory accessible via MMDC0, while TZASC2 provides protection for memory accessible via MMDC1.

Enabling TZASCs is expected to have a slight impact on memory performance. Exact value cannot be stated, since varies, depending on specific application software.

The proper and preferred method of enabling the TZASCs, is by burning the TZASC_ENABLE fuse.

For every power-up cycle, with TZASC_ENABLE fuse burned, the Boot ROM code will seamlessly handle the enable and engage of the TZASC modules and their clocks, leaving them in their active state, (i.e. enabled, and not bypassed). From this state and on, it is the responsibility of OS image, to configure the memory regions protection, per a specific application / use-case needs.

A configuration lock bits ("TZASC1_BOOT_LOCK", "TZASC2_BOOT_LOCK" in GPR3 register of IOMUXC) once set, will block any attempts to change the security settings, past the OS image configuration code settings. The configuration locking is in place, until the next hardware reset cycle.

NOTE

Engaging the TZASC functionality (i.e. - not bypassed), has to be done while DDR bus is guaranteed to be idle, with no pending transactions.

Enabling TZASC, without use of the associated fuse, is possible, but not trivial, since has to be done while traffic to DDR is guaranteed to be stopped. A typical way of achieving this, is by:

- Ensuring no other master can issue accesses to DDR.
- Protect against program flow change, to DDR space, (Disable interrupts, ans such).
- Run switching code from internal RAM.

TZASC enabling code has to handle the data-path mux control (via TZASC1_BYPASS, TZASC2_BYPASS, bits in GPR9, in IOMUXC module) as well as clock enabling (in LPCG module), and configuration locking, if desired (via set of TZASC1_BOOT_LOCK, TZASC2_BOOT_LOCK bits in GPR3, IOMUXC module).

The TZASC_BYPASS bit(s) in GPR9 register, once set, preserve their values until the next power-up cycle ("Sticky" type), in order to protect against unauthorized 'disable' operation.

63.4 Address Mapping in various memory mapping modes

The address configured to the TZASC controller(s) must match the "local addresses" as being passed on to the DDR controller(s).

The DDR controllers "local addresses" are referred to, as the addresses seen by the DDR controllers. In the single channel (single controller - x32 or x64) case, these addresses are identical to the physical addresses used in the system (0-4GB range), and are accessible via MMDC0.

However, for other, memory maps, such as 2x32 fixed and 2x32 Interleaved options, each MMDC and associated TZASC may be seeing different addresses. For SoC specific options, refer to DDR mapping to MMDC controller ports in the “Memory Maps” chapter.

For the 2x32 Fixed map (LPDDR2 only) , each MMDC (and TZASC) sees a 0-2GB address ranges.

For the "4KB address interleaved" scheme (i.e. LPDDR2 Dual channel 2x 32-bit, Interleaved mapping), addresses are split between MMDC (and TZASC) such that even 4K Bytes regions are mapped to MMDC0, while odd ones are mapped to MMDC1. Both regions are allocated in the 0-2GB space.

The memory regions configured in TZASC controllers ("base_address_low", "base_address_high" fields in "Region Setup Low/High" registers) must take into consideration the address translation done in 4KB interleaving scheme. In practice, the "local addresses" are composed of the original address, with address bit-12 (bit index 13) omitted from the address value, while higher bits "[31:13]" are shifter to the right one place, in place of bit-12.

Memory "aliasing" implications on TZASC settings - in systems which does not utilize the maximal supported DDR space the controller is designed for, the whole DDR memory map becomes "aliased" (replicated) by the size of the physical memory used. In such cases, the TZASC must be configured to protect all aliased regions as well (i.e. effectively reducing the number of available TZASC regions, since all aliased regions must be handled, for each "real" space needing protection).

For complete details on TZASC functionality and the programming model, see the ARM document, “CoreLink™ TrustZone Address Space Controller TZC-380 Technical Reference Manual, (Rev r0p1 or newer)”, available at **http://infocenter.arm.com**.

Chapter 64
Universal Asynchronous Receiver/Transmitter (UART)

64.1 Overview

Universal Asynchronous Receiver/Transmitter (UART) provides serial communication capability with external devices through a level converter and an RS-232 cable or through use of external circuitry that converts infrared signals to electrical signals (for reception) or transforms electrical signals to signals that drive an infrared LED (for transmission) to provide low speed IrDA compatibility.

UART supports NRZ encoding format , RS485 compatible 9 bit data format and IrDA-compatible infrared slow data rate (SIR) format.

The following figure is the UART block diagram.

The "Module Clock" is the UART_CLK which comes from CCM. The "Peripheral Clock" is the IPG_CLK which comes from CCM.

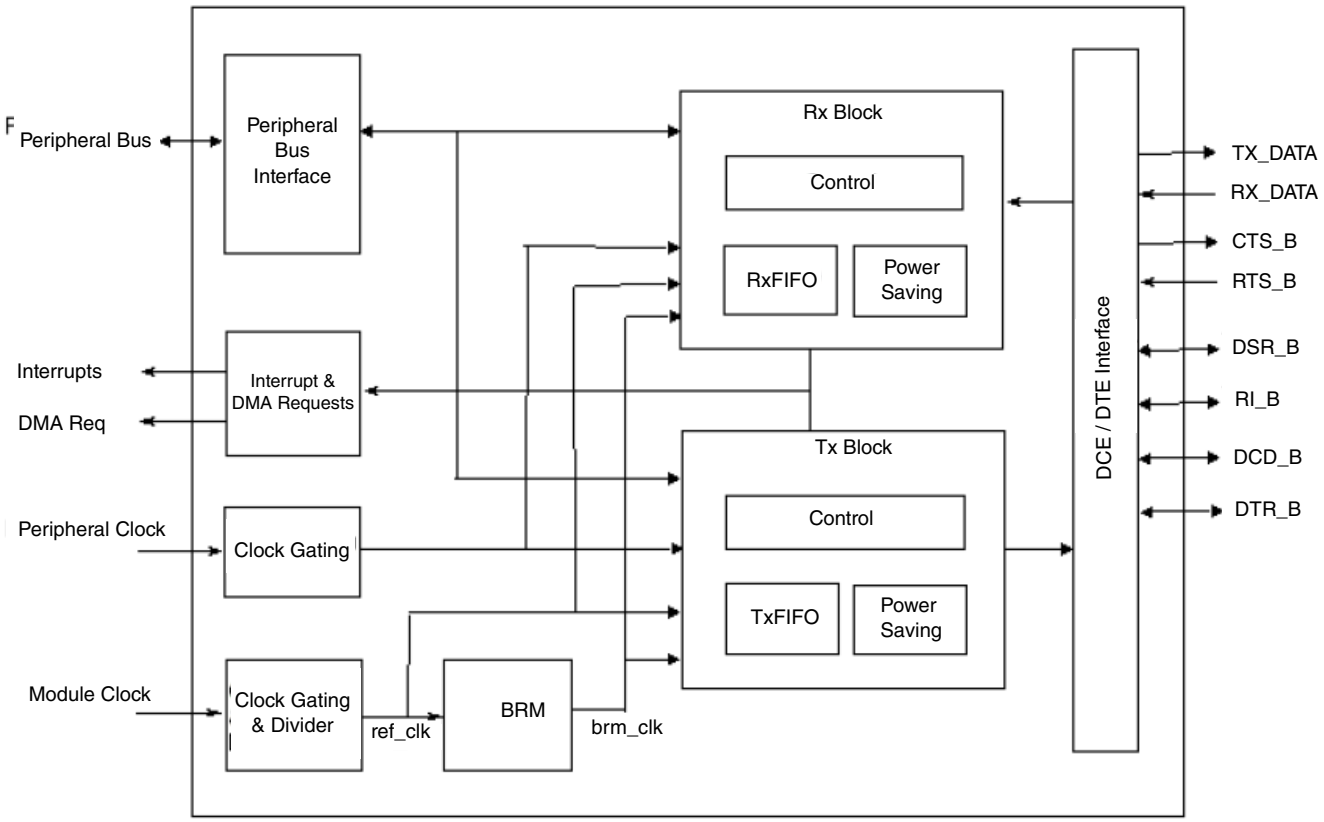


Figure 64-1. UART Block Diagram

64.1.1 Features

The UART includes the following features:

- High-speed TIA/EIA-232-F compatible, up to 5.0 Mbit/s
- Serial IR interface low-speed, IrDA-compatible (up to 115.2 Kbit/s)
- 9-bit or Multidrop mode (RS-485) support (automatic slave address detection)
- 7 or 8 data bits for RS-232 characters, or 9 bit RS-485 format
- 1 or 2 stop bits
- Programmable parity (even, odd, and no parity)
- Hardware flow control support for request to send (RTS_B) and clear to send (CTS_B) signals
- RS-485 driver direction control via CTS_B signal
- Edge-selectable RTS_B and edge-detect interrupts
- Status flags for various flow control and FIFO states
- Voting logic for improved noise immunity (16x oversampling)
- Transmitter FIFO empty interrupt suppression
- UART internal clocks enable/disable

- Auto baud rate detection (up to 115.2 Kbit/s)
- Receiver and transmitter enable/disable for power saving
- RX_DATA input and TX_DATA output can be inverted respectively in RS-232/ RS-485 mode
- DCE/DTE capability
- RTS_B, IrDA asynchronous wake (AIRINT), receive asynchronous wake (AWAKE), RI_B (DTE only), DCD_B (DTE only), DTR_B (DCE only) and DSR_B (DTE only) interrupts wake the processor from STOP mode
- Maskable interrupts
- Two DMA Requests (TxFIFO DMA Request and RxFIFO DMA Request)
- Escape character sequence detection
- Software reset (SRST_B)
- Two independent, 32-entry FIFOs for transmit and receive
- The peripheral clock can be totally asynchronous with the module clock. The module clock determines baud rate. This allows frequency scaling on peripheral clock (such as during DVFS mode) while remaining the module clock frequency and baud rate.

64.1.2 Modes of operation

- Serial RS-232NRZ mode
- 9-bit RS-485 mode
- IrDA mode

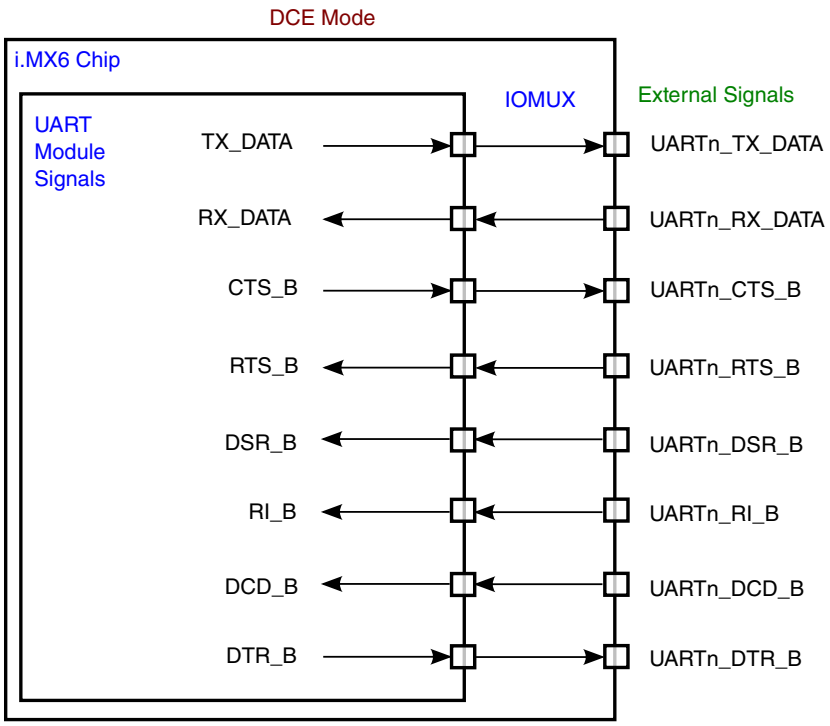
To set UART in different modes, see the table below.

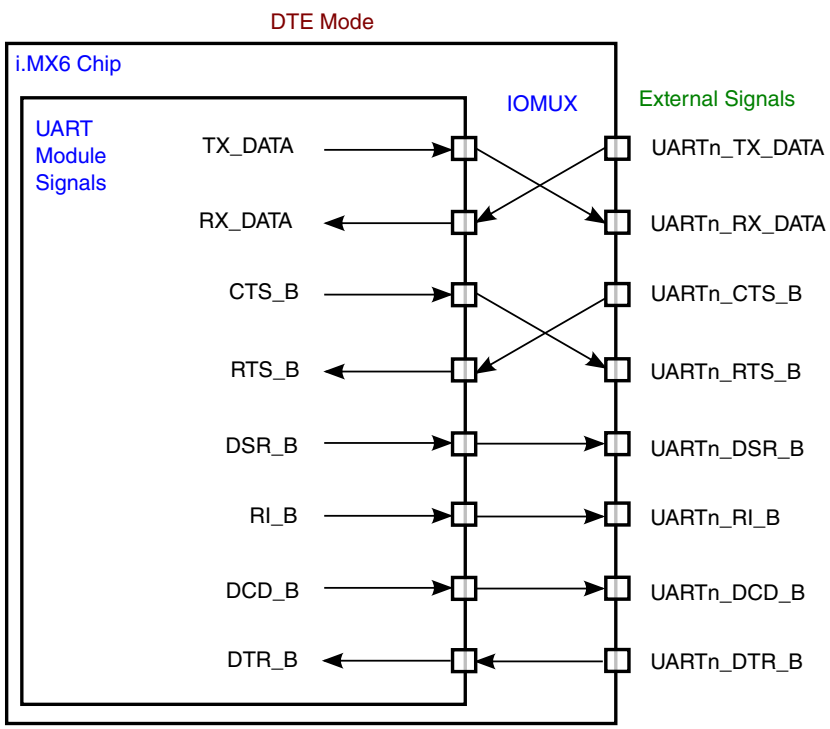
Table 64-1. UART mode definition

MDEN (UMCR[0])	IREN (UCR1[7])	UART Mode	Descriptioin
0	0	RS-232	RXD/TXD data is serial RS-232 NRZ format
0	1	IrDA (Interface)	RXD/TXD data is IrDA-compatible infrared slow data rate (SIR) format
1	0	RS-485	RXD/TXD data is RS485 compatible 9 bit data format
1	1	Undefined	Undefined

64.2 External Signals

The chip-level IOMUX modifies the direction and routing of the UART signals based on whether the UART is operating in DCE mode (UARTn_UFCR[DCEDTE]=0) or DTE mode (UARTn_UFCR[DCEDTE]=1). The routing of the external signals to the UART module is shown in the figure below.


DCE Mode
i.MX6 Chip
IOMUX
External Signals
UART
Module
Signals
TX_DATA
RX_DATA
CTS_B
RTS_B
DSR_B
RI_B
DCD_B
DTR_B
UARTn_TX_DATA
UARTn_RX_DATA
UARTn_CTS_B
UARTn_RTS_B
UARTn_DSR_B
UARTn_RI_B
UARTn_DCD_B
UARTn_DTR_B


DTE Mode
i.MX6 Chip
IOMUX
External Signals
UART
Module
Signals
TX_DATA
RX_DATA
CTS_B
RTS_B
DSR_B
RI_B
DCD_B
DTR_B
UARTn_TX_DATA
UARTn_RX_DATA
UARTn_CTS_B
UARTn_RTS_B
UARTn_DSR_B
UARTn_RI_B
UARTn_DCD_B
UARTn_DTR_B

mode

The following table describes the external signals of UART:

Table 64-2. UART1 External Signals

Signal	Description	Pad	Mode	Direction
UART1_CTS_B	Clear to send	EIM_D19	ALT4	O
		SD3_DAT0	ALT1	
UART1_DCD_B	Data carrier detected	EIM_D23	ALT3	IO
UART1_DSR_B	Data set ready	EIM_D25	ALT7	IO
UART1_DTR_B	Data terminal ready	EIM_D24	ALT7	IO
UART1_RI_B	Ring indicator	EIM_EB3	ALT3	IO
UART1_RTS_B	Request to send	EIM_D20	ALT4	I
		SD3_DAT1	ALT1	
UART1_RX_DATA	Serial / infrared data receive	CSI0_DAT11	ALT3	I
		SD3_DAT6	ALT1	
UART1_TX_DATA	Serial/infrared data transmit	CSI0_DAT10	ALT3	O
		SD3_DAT7	ALT1	

Table 64-3. UART2 External Signals

Signal	Description	Pad	Mode	Direction
UART2_CTS_B	Clear to send	EIM_D28	ALT4	O
		SD3_CMD	ALT1	
		SD4_DAT6	ALT2	
UART2_RTS_B	Request to send	EIM_D29	ALT4	I
		SD3_CLK	ALT1	
		SD4_DAT5	ALT2	
UART2_RX_DATA	Serial / infrared data receive	EIM_D27	ALT4	I
		GPIO_8	ALT4	
		SD3_DAT4	ALT1	
		SD4_DAT4	ALT2	
UART2_TX_DATA	Serial/infrared data transmit	EIM_D26	ALT4	O
		GPIO_7	ALT4	
		SD3_DAT5	ALT1	
		SD4_DAT7	ALT2	

Table 64-4. UART3 External Signals

Signal	Description	Pad	Mode	Direction
UART3_CTS_B	Clear to send	EIM_D23	ALT2	O
		EIM_D30	ALT4	

Table continues on the next page...

Table 64-4. UART3 External Signals (continued)

Signal	Description	Pad	Mode	Direction
		SD3_DAT3	ALT1	
UART3_RTS_B	Request to send	EIM_D31	ALT4	I
		EIM_EB3	ALT2	
		SD3_RST	ALT1	
UART3_RX_DATA	Serial / infrared data receive	EIM_D25	ALT2	I
		SD4_CLK	ALT2	
UART3_TX_DATA	Serial/infrared data transmit	EIM_D24	ALT2	O
		SD4_CMD	ALT2	

Table 64-5. UART4 External Signals

Signal	Description	Pad	Mode	Direction
UART4_CTS_B	Clear to send	CSI0_DAT17	ALT3	O
UART4_RTS_B	Request to send	CSI0_DAT16	ALT3	I
UART4_RX_DATA	Serial / infrared data receive	CSI0_DAT13	ALT3	I
		KEY_ROW0	ALT4	
UART4_TX_DATA	Serial/infrared data transmit	CSI0_DAT12	ALT3	O
		KEY_COL0	ALT4	

"The user must configure the input path to the UART by properly configuring the DAISY bits in the IOMUXC_UARTn_RX_DATA_INPUT and the IOMUXC_UARTn_UART_RTS_B_SELECT_INPUT registers.

64.2.1 Detailed Signal Descriptions

64.2.1.1 Interrupt Signals

64.2.1.1.1 *interrupt_uart* - UART Interrupt

Output interrupt request.

64.2.1.2 DMA Request Signals

64.2.1.2.1 *dma_req_rx* - Receiver DMA Request

Output DMA Request signal for receiver interface.

64.2.1.2.2 *dma_req_tx* - Transmitter DMA Request

Output DMA Request signal for transmitter interface. Set at 0 when TXDMAEN (UCR1[3]) is at 1 and TRDY (USR1[13]) is also at 1.

64.2.1.3 Special Signals

64.2.1.3.1 *stop_req* - Stop Mode

Input stop mode. Indicates to UART that ARM platform is going to enter in Stop Mode and clocks are going to stop running.

See Low Power Modes for more information about Stop Mode.

64.2.1.3.2 *doze_req* - Doze Mode

Input doze mode. ARM platform requests UART to switch in doze mode (power saving mode).

See Low Power Modes for more information about Doze Mode.

64.2.1.3.3 *debug_req* - Debug Mode

Input debug mode. Indicates UART it has to enter in debug mode.

See UART Operation in System Debug State, for more information about Debug Mode.

64.3 Clocks

The table found here describes the clock sources for UART.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 64-6. UART Clocks

Clock name	Clock Root	Description
ipg_clk	ipg_clk_root	Peripheral clock
ipg_clk_s	ipg_clk_root	Peripheral access clock
ipg_perclk	uart_clk_root	Module clock

64.4 Functional Description

This section provides a complete functional description of the block.

64.4.1 Interrupts and DMA Requests

See the following table for the lists of all interrupt and DMA signals and associated interrupt and DMA sources of the UART. See register description section for explanation of interrupt/DMA enable and status.

Table 64-7. Interrupts and DMA

Interrupt/DMA Output	Interrupt/DMA Enable	Enable Register Location	Interrupt/DMA Flag	Flag Register Location
interrupt_uart	RRDYEN	UCR1 (bit 9)	RRDY	USR1 (bit 9)
	IDEN	UCR1 (bit 12)	IDLE	USR2 (bit 12)
	DREN	UCR4 (bit 0)	RDR	USR2 (bit 0)
	RXDSEN	UCR3 (bit 6)	RXDS	USR1 (bit 6)
	ATEN	UCR2 (bit 3)	AGTIM	USR1 (bit 8)
interrupt_uart	TXMPTYEN	UCR1 (bit 6)	TXFE	USR2 (bit 14)
	TRDYEN	UCR1 (bit 13)	TRDY	USR1 (bit 13)
	TCEN	UCR4 (bit 3)	TXDC	USR2 (bit 3)
interrupt_uart	OREN	UCR4 (bit 1)	ORE	USR2 (bit 1)
	BKEN	UCR4 (bit 2)	BRCD	USR2 (bit 2)
	WKEN	UCR4 (bit 7)	WAKE	USR2 (bit 7)
	ADEN	UCR1 (bit 15)	ADET	USR2 (bit 15)
	ACIEN	UCR3 (bit 0)	ACST	USR2 (bit 11)
	ESCI	UCR2 (bit 15)	ESCF	USR1 (bit 11)
	ENIRI	UCR4 (bit 8)	IRINT	USR2 (bit 8)
	AIRINTEN	UCR3 (bit 5)	AIRINT	USR1 (bit 5)
	AWAKEN	UCR3 (bit 4)	AWAKE	USR1 (bit 4)
	FRAERREN	UCR3 (bit 11)	FRAERR	USR1 (bit 10)
	PARERREN	UCR3 (bit 12)	PARITYERR	USR1 (bit 15)
	RTSDEN	UCR1 (bit 5)	RTSD	USR1 (bit 12)
	RTSEN	UCR2 (bit 4)	RTSF	USR2 (bit 4)
	DTREN (DCE)	UCR3 (bit 13)	DTRF	USR2 (bit 13)
	RI (DTE)	UCR3 (bit 8)	RIDELT	USR2 (bit 10)
	DCD (DTE)	UCR3 (bit 9)	DCDDELT	USR2 (bit 6)
	DTRDEN	UCR3 (bit 3)	DTRD	USR1 (bit 7)

Table continues on the next page...

Table 64-7. Interrupts and DMA (continued)

Interrupt/DMA Output	Interrupt/DMA Enable	Enable Register Location	Interrupt/DMA Flag	Flag Register Location
	SADEN	UMCR (bit 3)	SAD	USR1 (bit 3)
dma_req_rx	RXDMAEN	UCR1 (bit 8)	RRDY	USR1 (bit 9)
	ATDMAEN	UCR1 (bit 2)	AGTIM	USR1 (bit 8)
	IDDMAEN	UCR4 (bit 6)	IDLE	USR2 (bit 12)
dma_req_tx	TXDMAEN	UCR1 (bit 3)	TRDY	USR1 (bit 13)

64.4.2 Clocks

This section describes clocks and special clocking requirements of the UART.

64.4.2.1 Clock requirements

UART module receives 2 clocks, *peripheral_clock* and *module_clock*. The *peripheral_clock* is used as write clock of the TxFIFO, read clock of the RxFIFO and synchronization of the modem control input pins. It must always be running when UART is enabled. There is an exception in stop mode (see Clocking in Low-Power Modes).

The *module_clock* is for all the state machines, writing RxFIFO, reading TxFIFO, etc. It must always be running when UART is sending or receiving characters.This clock is used in order to allow frequency scaling on *peripheral_clock* without changing configuration of baud rate (*module_clock* staying at a fixed frequency).

The constraints on *peripheral_clock* and *module_clock* are as follows:

- *peripheral_clock* and *module_clock* can totally be asynchronous. They can also be synchronous.
- Due to the 16x oversampling of the incoming characters, *module_clock* frequency must always be greater or equal to 16x the maximum baud rate. For example, if max baud rate is 4 Mbit/s, *module_clock* must be greater or equal to 4 M x 16 = 64MHz.

NOTE

The restriction that *peripheral_clock* frequency must be higher or equal to 16x baud rate has been removed. There is no limitation on *peripheral_clock* frequency to baud rate.

64.4.2.2 Maximum Baud Rate

The max baud rate the UART can support is determined by the max frequency of the module_clock.

For example, if the SoC can provide the fastest module_clock 66.5 MHz, the UART can transmit and receive serial data with the maximum baud rate 66.5M/16 = 4.15 Mbit/s.

The UART supports serial IR interface low speed. In the low speed IrDA mode, the max baud rate is 115.2Kbit/s. To support the 115.2Kbit/s, module_clock frequency must be higher or equal to 1.8432MHz.

64.4.2.3 Clocking in Low-Power Modes

The UART supports 2 low-power modes: DOZE and STOP.

In STOP mode (input pin *stop_req* is at '1'), the UART doesn't need any clock. In this mode the UART can wake-up the ARM platform with the asynchronous interrupts (see Low Power Modes).

- If before entering in STOP mode the software has enabled RTSDEN interrupt, when RTS will change state (put at '0' by external device started to send), the asynchronous interrupt will wake-up the system, *peripheral_clock* and *module_clock* will be provided to the UART before first start bit, so that no data will be lost.
- If RTS doesn't change state (already at '0' before entering in STOP mode), then wake-up interrupt (AWAKE) will be sent at the arrival of first Start bit (on falling edge). In this case, the UART must receive the *peripheral_clock* and *module_clock* during the first half of start bit to correctly receive this character (for example, at 115.2 Kbit/s, UART must receive *peripheral_clock* and *module_clock* at maximum 4.3 microseconds after falling edge of Start bit). If the UART receives *peripheral_clock* and *module_clock* too late, first character will be lost, and so should be dropped. Also, if autobaud detection is enabled, the first character won't be correctly received and another autobaud detection will need to be initiated.

In Doze mode, UART behavior is programmable through DOZE bit (UCR1[1]). If DOZE bit is set to '1', then UART is disabled in Doze mode, and in consequence, UART clocks can be switched-off (after being sure UART is not transmitting nor receiving). On the contrary, if DOZE bit is set to '0', UART is enabled and it must receive *peripheral_clock* and *module_clock* .

64.4.3 General UART Definitions

Definitions of terms that occurs the following discussions are given in this section.

- Bit Time-The period of time required to serially transmit or receive 1 bit of data (1 cycle of the baud rate frequency).
- Start bit-The bit time of a logic 0 that indicates the beginning of a data frame. A start bit begins with a 1-to-0 transition, and is preceded by at least 1 bit time of logic 1.
- Stop bit-1 bit time of logic 1 that indicates the end of a data frame.
- BREAK-A frame in which all of the data bits, including the stop bit, are logic 0. This type of frame is usually sent to signal the end of a message or the beginning of a new message.
- Mark - When no data is being sent, the serial port's transmit pin's voltage is 1 and is said to be in a MARK state.
- Space - The serial port can also be forced to keep the transmit pin at a 0 and is said to be the SPACE or BREAK state.
- Frame-A start bit followed by a specified number of data or information bits and terminated by a stop bit. The number of data or information bits depends on the format specified and must be the same for the transmitting device and the receiving device. The most common frame format is 1 start bit followed by 8 data bits (least significant bit first) and terminated by 1 stop bit. An additional stop bit and a parity bit also can be included.
- Framing Error-An error condition that occurs when the stop bit of a received frame is missing, usually when the frame boundaries in the received bit stream are not synchronized with the receiver bit counter. Framing errors can go undetected if a data bit in the expected stop bit time happens to be a logic 1. A framing error is always present on the receiver side when the transmitter is sending BREAKs. However, when the UART is programmed to expect 2 stop bits and only the first stop bit is received, this is not a framing error by definition.
- Parity Error-An error condition that occurs when the calculated parity of the received data bits in a frame does not match the parity bit received on the RX_DATA input. Parity error is calculated only after an entire frame is received.
- Idle-One in NRZ encoding format and selectable polarity in IrDA mode.
- Overrun Error-An error condition that occurs when the latest character received is ignored to prevent overwriting a character already present in the UART receive buffer (RxFIFO). An overrun error indicates that the software reading the buffer (RxFIFO) is not keeping up with the actual reception of characters on the RX_DATA input.

64.4.3.1 RTS_B - UART Request To Send

The UART Request To Send input controls the transmitter. The modem or other terminal equipment signals the UART when it is ready to receive by setting '0' on the RTS_B pin.

Normally, the transmitter waits until this signal is active (low) before transmitting a character, however when the Ignore RTS (IRTS) bit is set, the transmitter sends a character as soon as it is ready to transmit. An interrupt (RTSD) can be posted on any transition of this pin and can wake the ARM platform from STOP mode on its assertion. When RTS_B is set to '1' during a transmission, the UART transmitter finishes transmitting the current character and shuts off. The contents of the TxFIFO (characters to be transmitted) remain undisturbed. The operation of this input is the same regardless of whether the UART is in DTE or DCE mode.

64.4.3.2 RTS Edge Triggered Interrupt

The input to the RTS_B pin can be programmed to generate an interrupt on a selectable edge.

See the table below for summary of the operation of the RTS edge triggered interrupt (RTSF).

To enable the RTS_B pin to generate an interrupt, set the request to send interrupt enable (RTSEN) bit (UCR2[4]) to 1. Writing 1 to the RTS_B edge triggered interrupt flag (RTSF) bit (USR2[4]) clears the interrupt flag. The interrupt can occur on the rising edge, falling edge, or either edge of the RTS_B input. The request to send edge control (RTEC) field (UCR2[10:9]) programs the edge that generates the interrupt. When RTEC is set to 0x00 and RTSEN = 1, the interrupt occurs on the rising edge (default). When RTEC is set to 0x01 and RTSEN = 1, the interrupt occurs on the falling edge. When RTEC is set to 0x1X and RTSEN = 1, the interrupt occurs on either edge. This is a synchronous interrupt. The RTSF bit is cleared by writing 1 to it. Writing 0 to RTSF has no effect.

Table 64-8. RTS_B Edge Triggered Interrupt Truth Table

RTS_B	RTSEN	RTEC [1]	RTEC [0]	RTSF	Interrupt Occurs On…	interrupt_uart
X	0	X	X	0	Interrupt disabled	1
1->0	1	0	0	0	Rising edge	1
0->1	1	0	0	1	Rising edge	0
1->0	1	0	1	1	Falling edge	0
0->1	1	0	1	0	Falling edge	1
1->0	1	1	X	1	Either edge	0
0->1	1	1	X	1	Either edge	0

There is another RTS_B interrupt that is not programmable. The status bit RTSD asserts the *interrupt_uart* interrupt when the RTS_B delta interrupt enable = 1. This is an asynchronous interrupt. The RTSD bit is cleared by writing 1 to it. Writing 0 to the RTSD bit has no effect.

64.4.3.3 DTR_B - Data Terminal Ready

This signal indicates the general readiness of the Data Terminal Equipment (DTE). This signal is an input in DCE mode and an output in DTE mode. If the connection between the DCE and the DTE is established once, the DTR_B signal must remain active throughout the whole connection time.

In general the DTR_B and DSR_B signals are responsible for establishing the connection. RTS_B and CTS_B are responsible for the data transfer and the transfer direction in the case of a half-duplex configuration. The DTR_B signal is like a "main switch". If the DTR_B signal is inactive the RTS_B and CTS_B signals have no effect. In DCE mode, an interrupt (DTRD) can be posted on any transition of this pin and can wake the ARM platform from STOP mode on its assertion.

64.4.3.4 DSR_B - Data Set Ready

This signal indicates the general readiness of the DCE. This signal is an output in DCE mode and an input in DTE mode. The DCE uses this signal to inform the DTE that it is switched on, has completed all preparations and can communicate with the DTE.

In DTE mode, an interrupt (DTRD) can be posted on any transition of this pin and can wake the ARM platform from STOP mode on its assertion.

64.4.3.5 DTR_B/DSR_B Edge Triggered Interrupt

The DTR_B input pin (DCE mode) or DSR_B input pin (DTE mode) can be configured to cause an interrupt on a selectable edge.

See the table below for summary of the operation of the DTR/DSR edge triggered interrupt. To enable the interrupt, set the DTREN bit (UCR3[13]) to '1'. Write a "one" to the DTRF bit (USR2[13]) to clear the interrupt flag.

The interrupt can be configured to occur on either the rising, falling, or either edge of the DTR_B/DSR_B input. Write to the DPEC[1:0] bits (UCR3[15:14]) to program which edge will cause an interrupt. If the bits are set to 00b and DTREN = 1, the interrupt will

occur on the rising edge (default). If the bits are set to 01b and DTREN = 1, the interrupt will occur on the falling edge. If the bits are set to 1Xb and DTREN = 1, the interrupt will occur on either edge.

Table 64-9. DTR/DSR_B Edge Triggered Interrupt Truth Table

DTR_B / DSR_B	DTREN	DPEC[1]	DPEC[0]	DTRF	Interrupt occurs on:	interrupt_uart
X	0	X	X	0	turned off	1
1->0	1	0	0	0	rising edge	1
0->1	1	0	0	1	rising edge	0
1->0	1	0	1	1	falling edge	0
0->1	1	0	1	0	falling edge	1
1->0	1	1	X	1	either edge	0
0->1	1	1	X	1	either edge	0

64.4.3.6 DCD_B - Data Carrier Detect

This signal is an output in DCE mode and an input in DTE mode. If used, the DCE device uses this signal to inform the DTE it has detected the carrier signal and the connection will be set up. This signal remains active while the connection remains established.

In DTE mode this input can trigger an interrupt on changing state. This is achieved by setting to '1' the interrupt enable bit (DCD, UCR3[9]). The change state is reflected in DCDDELT (USR2[6]). Also, the state of the Data Carrier Detect input is mirrored in the status register DCDIN (USR2[5]).

64.4.3.7 RI_B - Ring Indicator

This signal is an output in DCE mode and an input in DTE mode. If used, the DCE device uses this signal to inform the DTE that a ring just occurred.

In DTE mode this input can trigger an interrupt on changing state. This is achieved by setting to '1' the interrupt enable bit (RI, UCR3[8]). The change state is reflected in RIDELT (USR2[10]). Also, the state of the Ring Indicator input is mirrored in the status register RIIN (USR2[9]).

64.4.3.8 CTS_B - Clear To Send

This output pin serves two purposes. Normally, the receiver indicates that it is ready to receive data by asserting this pin (low). When the CTS_B trigger level is programmed to trigger at 32 characters received and the receiver detects the valid start bit of the 33 character, it de-asserts this pin. The operation of this output is the same regardless of whether the UART is in DTE or DCE mode.

64.4.3.9 Programmable CTS_B Deassertion

The CTS_B output can also be programmed to deassert when the RxFIFO reaches a certain level. Setting the CTS trigger level (UCR4[15:10]) at any value less than 32 deasserts the CTS_B pin on detection of the valid start bit of the N + 1 character (where N is the trigger level setting). However, the receiver continues to receive characters until the RxFIFO is full.

64.4.3.10 TX_DATA - UART Transmit

This is the transmitter serial output. When operating in RS-232/RS-485 mode, NRZ encoded data is transmitted, and the data can be inverted (controlled by INVT (UCR3[1])) before transmitted. When operating in infrared mode, a 3/16 bit-period pulse is output for each 0 bit transmitted, and no pulse is output for each 1 bit transmitted.

For RS-232/RS-485 applications, this pin must be connected to an RS-232/RS-485 transmitter. The operation of this output is the same regardless of whether the UART is in DTE or DCE mode. See Figure 64-3.

64.4.3.11 RX_DATA - UART Receive

This is the receiver serial input. When operating in RS-232/RS-485 mode, NRZ encoded data is expected, and the data can be inverted (controlled by INVR (UCR4[9])) before sampled. When operating in infrared mode, a narrow pulse is expected for each 0 bit received and no pulse is expected for each 1 bit received.

External circuitry must convert the IR signal to an electrical signal. RS-232/RS-485 applications require an external RS-232/RS-485 receiver to convert voltage levels. The operation of this input is the same regardless of whether the UART is in DTE or DCE mode. See the figure below.

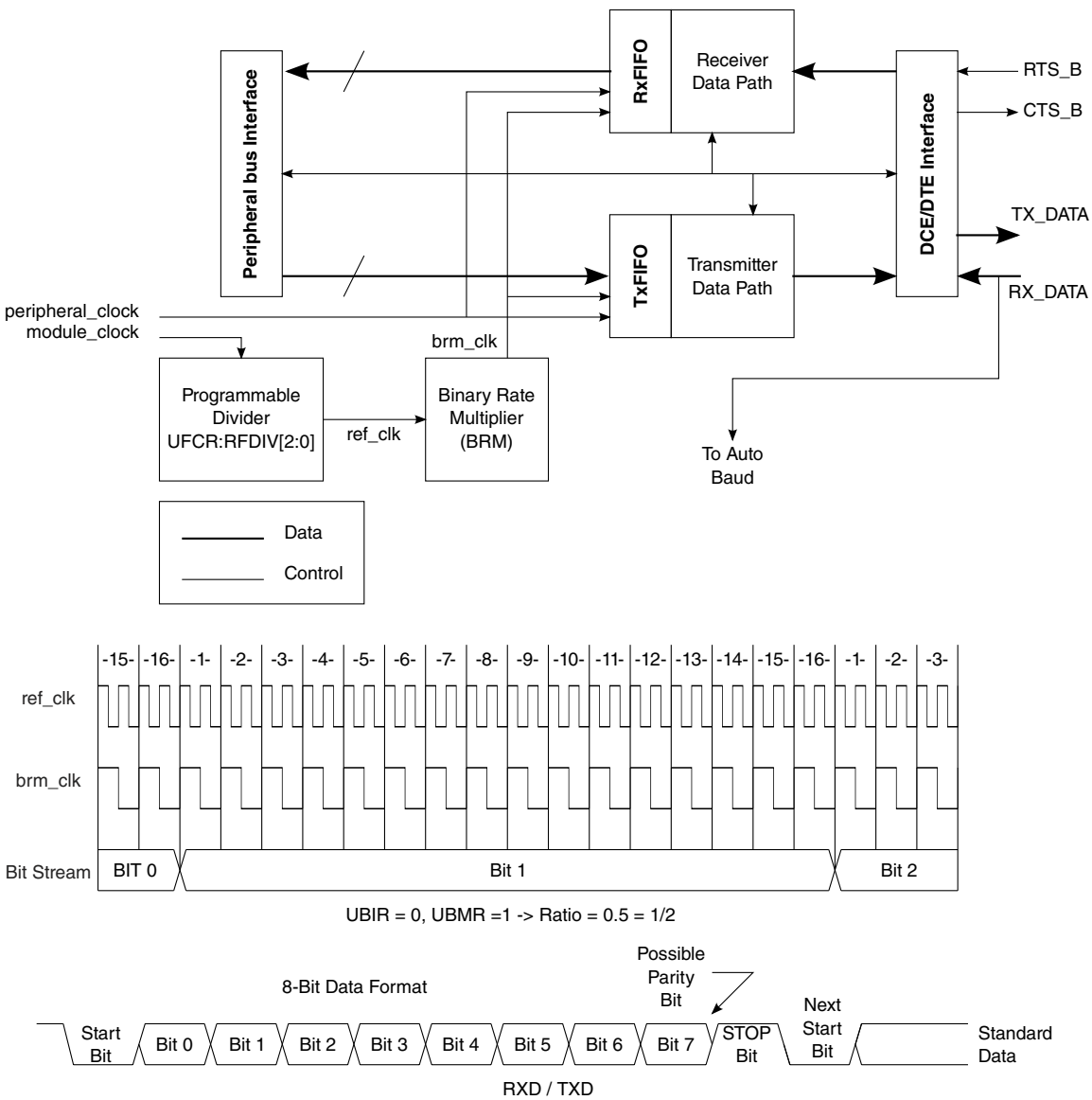


Figure 64-3. UART Simplified Block and Clock Generation Diagrams

64.4.4 Transmitter

The transmitter accepts a parallel character from the ARM platform and transmits it serially. The start, stop, and parity (when enabled) bits are added to the character.

When the ignore RTS bit (IRTS) is set, the transmitter sends a character as soon as it is ready to transmit. RTS_B can be used to provide flow-control of the serial data. When RTS_B is set to '1', the transmitter finishes sending the character in progress (if any), stops, and waits for RTS_B to be set to '0' again. Generation of BREAK characters and parity errors (for debugging purposes) is supported. The transmitter operates from the clock provided by the Binary Rate Multiplier(BRM). Normal NRZ encoded data is transmitted when the IR interface is disabled.

The transmitter FIFO (TxFIFO) contains 32 bytes. The data is written to TxFIFO by writing to the UTXD register with the byte data to the [7:0] bits. The data is written consecutively if the TxFIFO is not full. It is read (internally) consecutively if the TxFIFO is not empty. TXFULL bit (UTS[4]) can be used to control whether TXFIFO is full or not. The TxFIFO can be written regardless of the transmitter is disabled or enabled. If the UART is disabled, user can still write data into the TxFIFO correctly. But in this case the write access will yield to a transfer error.

64.4.4.1 Transmitter FIFO Empty Interrupt Suppression

The transmitter FIFO empty interrupt suppression logic suppresses the TXFE interrupt between writes to the TxFIFO.

When TxFIFO is empty, the software can either send one or several characters. If the software sends one character, it would write the character into the UTXD register, then that character is immediately transferred to the transmitter shift register, assuming the transmitter is already enabled. Without interrupt suppression logic, the TXFE interrupt flag would be set immediately. But, with this logic, the interrupt flag is set when the last bit of the character has been transmitted, for example, before the transmission of the parity bit (if exists) and the stop bit(s).

So, the suppression logic doesn't immediately send the TXFE interrupt flag. It allows the software to write another character to the TxFIFO before the interrupt flag is asserted.

When the transmitter shift register empties before another character is written to the TxFIFO, the interrupt flag is asserted. Writing data to the TxFIFO would release the interrupt flag. The interrupt flag is asserted on the following conditions:

- System Reset

- UART software reset
- When a single character has been written to Transmitter FIFO and then the Transmitter FIFO and the Transmitter Shift Register become empty until another character is written to the Transmitter FIFO
- The last character in the TxFIFO is transferred to the shift register, when TxFIFO contains two or more characters. See the figure below.

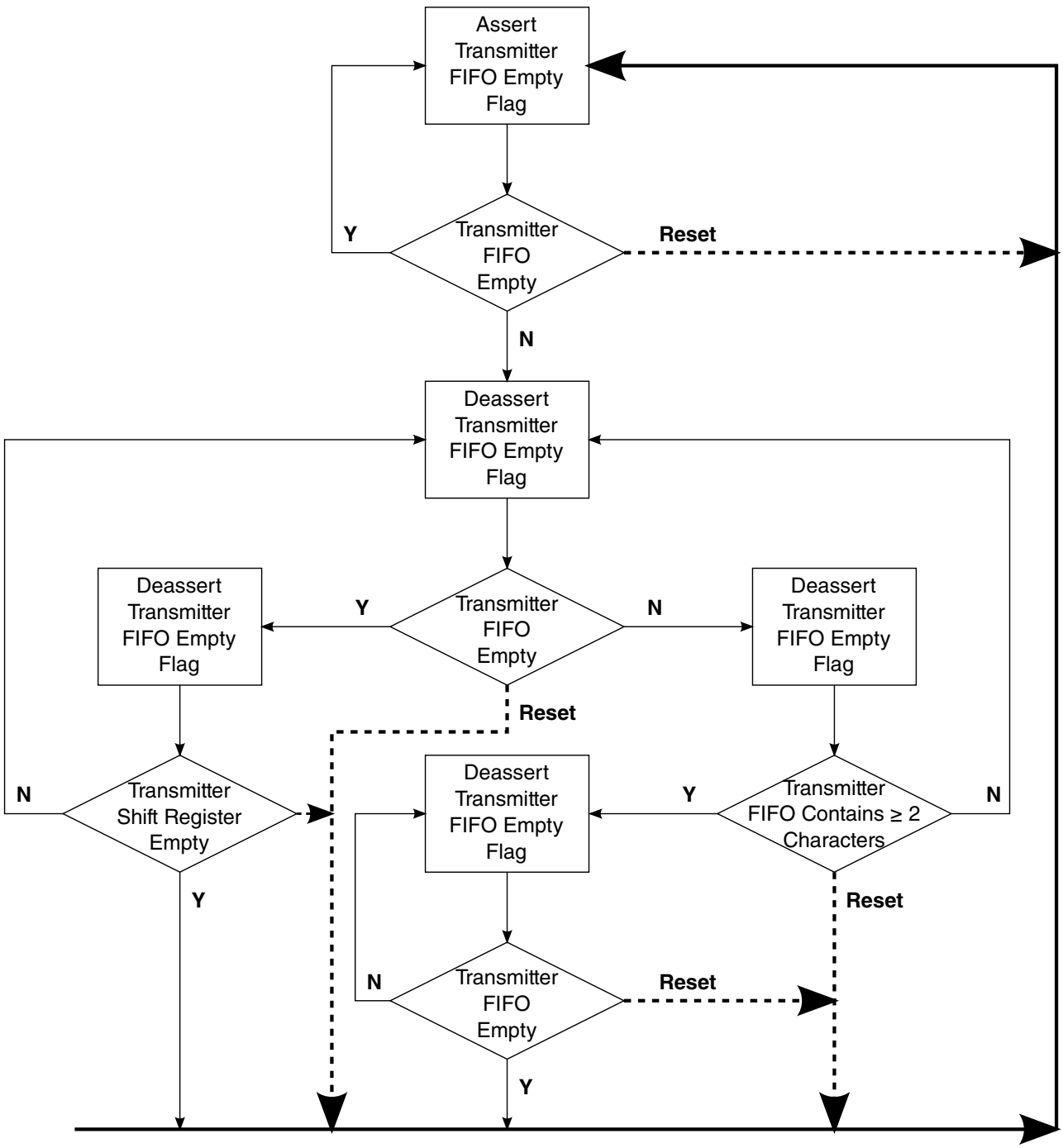


Figure 64-4. Transmitter FIFO Empty Interrupt Suppression Flow Chart

64.4.4.2 Transmitting a Break Condition

Asserting SNDBRK bit of the UCR1 Register forces the transmitter to send a break character (continuous zeros). The transmitter will finish sending the character in progress (if any) before sending break until this bit is reset.

The user is responsible to ensure that this bit is high for long enough to generate a valid BREAK. The transmitter samples SNDBRK after every bit is transmitted. Following completion of the BREAK transmission, the UART will transmit two mark bits. The user can continue to fill the FIFO and any character remaining will be transmitted when the break is terminated.

64.4.5 Receiver

See the figure below for the receiver flow chart.

The receiver accepts a serial data stream and converts it into parallel characters. When enabled, it searches for a start bit, qualifies it, and samples the following data bits at the bit-center.

Jitter tolerance and noise immunity are provided by sampling at a 16x rate and using voting techniques to clean up the samples. Once the start bit is found, the data bits, parity bit (if enabled), and stop bits (either 1 or 2 depending on user selection) are shifted in. Parity is checked and its status reported in the URXD register when parity is enabled. Frame errors and BREAKs are also checked and reported. When a new character is ready to be read by the ARM platform from the RxFIFO, the receive data ready (RDR = USR2[0]) bit is asserted and an interrupt is posted (if DREN = UCR4[0] = 1). If the receiver trigger level is set to 2 (RXTL[5:0] = UFCR[5:0] = 2), and 2 chars have been received into RxFIFO, the receiver ready interrupt flag (RRDY = USR1[9]) is asserted and an interrupt is posted if the receiver ready interrupt enable bit is set (RRDYEN = UCR1[9] = 1). If the UART Receiver Register (URXD) is read once, and in consequence there is only 1 character in the RxFIFO, the interrupt generated by the RDR bit is automatically cleared. The RRDY bit is cleared when the data in the RxFIFO falls below the programmed trigger level.

Normal NRZ encoded data is expected when the IR interface is disabled. The RxFIFO contains 32 half-word entries. Characters received are written consecutively into this FIFO. If the FIFO is full and a 33rd characters is received, this character will be ignored and the USR2[ORE] bit will be set.

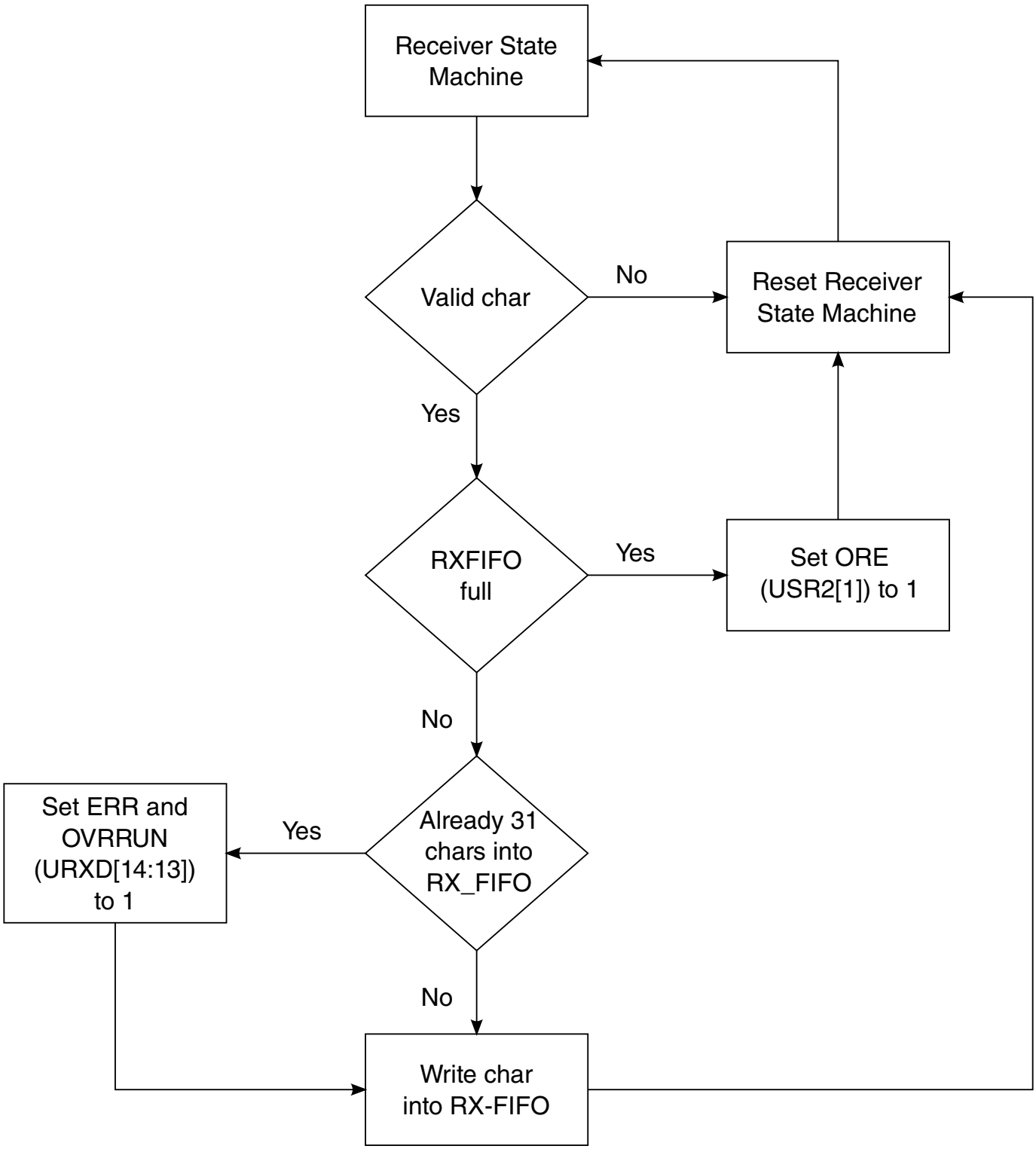


Figure 64-5. Receiver Flow Chart

64.4.5.1 Idle Line Detect

The receiver logic block includes the ability to detect an idle line. Idle lines indicate the end or the beginning of a message.

For an idle condition to occur:

- RxFIFO must be empty and
- RX_DATA pin must be idle for more than a configured number of frames (ICD[1:0] = UCR1[11:10]).

When the idle condition detected interrupt enable (IDEN = UCR1[12]) is set and the line is idle for 4 (default), 8, 16, or 32 (maximum) frames, the detection of an idle condition flags an interrupt (see the table below). When an idle condition is detected, the IDLE (USR2[12]) bit is set. Clear the IDLE bit by writing 1 to it. Writing 0 to the IDLE bit has no effect.

Table 64-10. Detection Truth Table

IDEN	ICD [1]	ICD [0]	IDLE	*interrupt_uart*
0	X	X	0	1
1	0	0	asserted after 4 idle frames	asserted after 4 idle frames
1	0	1	asserted after 8 idle frames	asserted after 8 idle frames
1	1	0	asserted after 16 idle frames	asserted after 16 idle frames
1	1	1	asserted after 32 idle frames	asserted after 32 idle frames
NOTE: This table assumes that no other interrupt is set at the same time this interrupt is set for the *interrupt_uart* signal. This table shows how this interrupt affects the *interrupt_uart* signal.				

During a normal message there is no idle time between frames. When all of the information bits in a frame are logic 1s, the start bit ensures that at least one logic 0 bit time occurs for each frame so that the IDLE bit is not asserted.

64.4.5.2 Aging Character Detect

The receiver block also includes the possibility to detect when at least one character has been sitting into the RxFIFO for a time corresponding to 8 characters. This aging character capability allows the UART to inform the ARM platform that there is less character into the RxFIFO than the Rx trigger and, no new character has been detected on the RXD line.

The aging capability is a timer which starts to count as soon as the RxFIFO is not empty and its trigger level is not reached (RRDY=0). This counter is reset when either a RxFIFO read is performed or another character starts to present on the RXD line. If none of those two events occurs, the bit AGTIM (USR1[8]) is set when the counter has

measured a time corresponding to 8 characters. AGTIM is cleared by writing a 1 to it. AGTIM can flag an interrupt to ARM platform on *interrupt_uart* if ATEN (UCR2[3]) has been set.

To summarize, AGTIM is set when:

- There is at least one character into RxFIFO.
- No read has occurred on RxFIFO and RXD line has stayed high, for a time corresponding to 8 characters.
- The RxFIFO trigger is not reached (RRDY=0)

64.4.5.3 Receiver Wake

The WAKE bit (USR2[7]) is set when the receiver detects a qualified Start bit. For this, two conditions must be fulfilled, firstly a falling edge on RX_DATA line must be detected and secondly the RX_DATA line must stays at low level for more than a half-bit duration.

When the wake interrupt enable WKEN (UCR4[7]) bit is enabled, the receiver flags an interrupt (*interrupt_uart*) if the WAKE status bit is set. The WAKE bit is cleared by writing 1 to it. Writing 0 to the WAKE bit has no effect. The WAKE status bit can be asserted in either serial RS-232 mode or IR mode. The generation of the WAKE interrupt needs the clock *module_clock* .

When the asynchronous wake interrupt (AWAKE) is enabled (AWAKEN = UCR3[4] = 1), and the ARM platform is in STOP mode, and UART clocks have been shut-off, then a falling edge detected on the receive pin (RX_DATA) asserts the AWAKE bit (USR1[4]) and the *interrupt_uart* interrupt to wake the ARM platform from STOP mode. Re-enable UART clocks and clear the AWAKE bit by writing 1 to it. Writing 0 to the AWAKE bit has no effect. When IR interface is enabled (UCR1[7]=1), the AWAKE bit is always not asserted. The generation of the asynchronous AWAKE interrupt does not need any clocks.

In IR mode, if the asynchronous IR WAKE interrupt is enabled (AIRINTEN = UCR3[5] = 1), and if the ARM platform is in STOP mode (UART clocks are off when ARM platform in STOP mode), then the detection of a falling edge on the receive pin (RXD_IR), asserts the AIRINT bit (USR1[5]), and the *interrupt_uart* interrupt. This interrupt wakes the ARM platform from STOP mode. Software re-enables UART clocks and clear the AIRINT bit by writing 1 to it. Writing 0 to the AIRINT bit has no effect. When IR interface is disabled (UCR1[7]=0), the AIRINT bit is always not asserted. The generation of the asynchronous AIRINT interrupt does not need any clocks.

Recommended procedure for programming the asynchronous interrupts is to first clear them by writing 1 to the appropriate bit in the UART Status Register 1 (USR1). Poll or enable the interrupt for the Receiver IDLE Interrupt Flag (RXDS) in the USR1. When asserted, the RXDS bit indicates to the software that the receiver state machine is in the idle state, the next state is idle, and the RX_DATA pin is idle (high). After following this procedure, enable the asynchronous interrupt and enter STOP mode.

64.4.5.4 Receiving a BREAK Condition

A BREAK condition is received when the receiver detects all 0s (including a 0 during the bit time of the stop bit) in a frame. The BREAK condition asserts the BRCD bit (USR2[2]) and writes only the first BREAK character to the RxFIFO. Clear the BRCD bit by writing 1 to it. Writing 0 to the BRCD bit has no effect.

Asserting BRCD would generate an interrupt on *interrupt_uart*. The interrupt generation can be masked using the control bit BKEN (UCR4[2]). Receiving a break condition will also effect the following bits in the receiver register URXD:

URXD(11) = BRK. While high this bit indicates that the current char was detected as a break.

URXD(12) = FRMERR. The frame error bit will always be set when BRK is set.

URXD(10) = PRERR. If odd parity was selected the parity error bit will also be set when BRK is set.

URXD(14) = ERR. The error detect bit indicates that the character present in the rx data field has an error status. This can be asserted by a break.

64.4.5.5 Vote Logic

The vote logic block provides jitter tolerance and noise immunity by sampling with respect to a 16x clock (*brm_clk*) and using voting techniques to clean up the samples. The voting is implemented by sampling the incoming signal constantly on the rising edge of the *brm_clk*.

See Figure 64-6. The receiver is provided with the majority vote value, which is 2 out of the 3 samples. For examples of the majority vote results of the vote logic, see the following table.

Table 64-11. Majority Vote Results

Samples	Vote
000	0
101	1
001	0
111	1

The vote logic captures a sample on every rising edge of *brm_clk*, however the receiver uses 16x oversampling to take its value in the middle of the sample character.

The receiver starts to count when the Start bit is set however it does not capture the contents of the RxFIFO at the time the Start bit is set. The start bit is validated when 0s are received for 7 consecutive 1/16 of bit times following the 1-to-0 transition. Once the counter reaches 0xF, it starts counting on the next bit and captures it in the middle of the sampling frame (see Table 64-11). All data bits are captured in the same manner. Once the stop bit is detected, the receiver shift register (SIPO_OUT) data is parallel shifted to the RxFIFO.

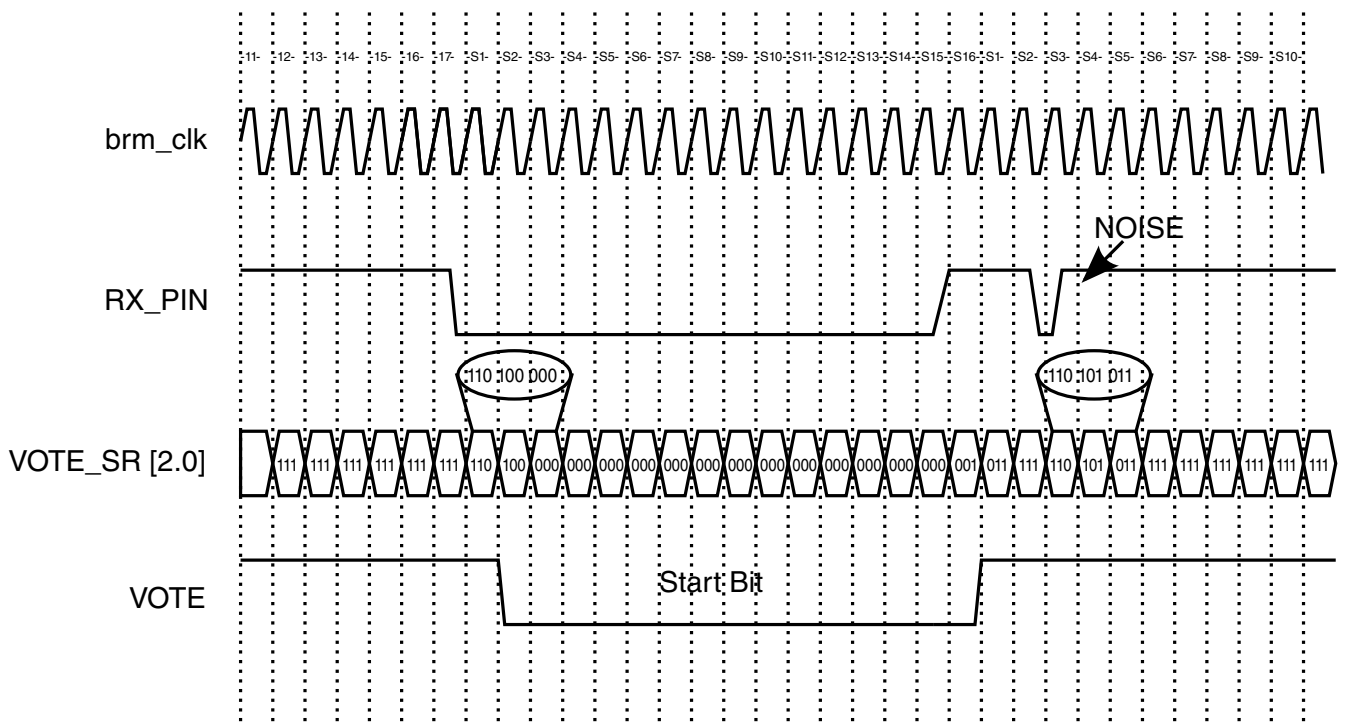


Figure 64-6. Majority Vote Results

A new feature has been recently implemented, it allows to re-synchronize the counter on each edge of RX_DATA line. This is automatic and allows to improve the immunity of UART against signal distortion.

There is a special case when the *brm_clk* frequency is too low and is unable to capture a 0 pulse in IrDA. In this case, the software must set the IRSC (UCR4[5]) bit so that the reference clock (after internal divider) is used for the voting logic. The pulse is validated by counting the length of the pulse.

Refer to Infrared Interface for more details.

64.4.5.6 Baud Rate Automatic Detection Logic

When the baud rate automatic detection logic is enabled, the UART locks onto the incoming baud rate. To enable this feature, set the automatic detection of baud rate bit (ADBR = UCR1[14] = 1) and write 1 to the ADET bit (USR2[15]) to clear it.

When ADET=0 and ADBR =1, the detection starts. Then, once the beginning of start bit (transition from 1-to-0 of RX_DATA) has been detected, UART starts a counter (UBRC) working at reference frequency. Once the end of start bit is detected (transition from 0-to-1 of RX_DATA), the value of UBRC - 1 is directly copied into UBMR register. UBIR register is filled with 0x000F.

So, at the end of start bit, registers gets following values:

```
UBRC = number of reference clock periods (after divider) during Start bit.
UBIR = 0x000F
UBMR = UBRC - 1
```

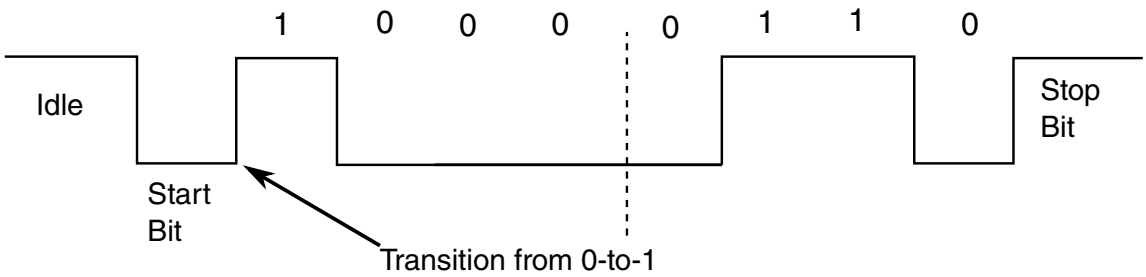
The updated values of the 3 registers can be read.

See Table 64-12 for list of parameters for baud rate detection and Figure 64-7 for baud rate detection protocol diagram.

If any of the UART BRM registers are simultaneously written by the baud rate automatic detection logic and by the peripheral data bus, the peripheral data bus would have lower priority.

Table 64-12. Baud Rate Automatic Detection

ADBR	ADET	Baud Rate Detection	*interrupt_uart*
0	X	Manual Configuration	1
1	0	Auto Detection Started	1
1	1	Auto Detection Complete	0
NOTE: This table assumes that no other interrupt is set at the same time this interrupt is set for the *interrupt_uart* signal.			



Note: LSB Transmitted first.

Figure 64-7. Baud Rate Detection Protocol Diagram

64.4.5.6.1 Baud Rate Automatic Detection Protocol

The receiver must receive an ASCII character "A" or "a" to verify proper detection of the incoming baud rate. When an ASCII character "A" (0x41) or "a" (0x61) is received and no error occurs, the Automatic Detect baud rate bit is set (ADET=1) and if the interrupt is enabled (ADEN=UCR1[15]=1), an interrupt *interrupt_uart* is generated.

When an ASCII character "A" or "a" is not received (because of a bit error or the reception of another character), the auto detection sequence restarts and waits for another 1-to-0 transition.

As long as ADET = 0 and ADBR = 1, the UART continues to try to lock onto the incoming baud rate. Once the ASCII character "A" or "a" is detected and the ADET bit is set, the receiver ignores the ADBR bit and continues normal operation with the calculated baud rate.

The UART interrupt is active (*interrupt_uart* = 0) as long as ADET = 1 and ADBR = 1. This can be disabled by clearing the automatic baud rate detection interrupt enable bit (ADEN = 0). Before starting an automatic baud rate detection sequence, set ADET = 0 and ADBR = 1.

The RxFIFO must contain the ASCII character "A" or "a" following the automatic baud rate detection interrupt.

The 16-bit UART Baud Rate Count Register (UBRC) is reset to 4 and stays at 0xFFFF when an overflow occurs. The UBRC register counts (measures) the duration of start bit. When the start bit is detected and counted, the UART Baud Rate Count Register retains its value until the next automatic baud rate detection sequence is initiated.

The Baud Rate Count Register counts only when auto detection is enabled.

64.4.5.6.2 New Baud Rate Determination

In order to fight against the problems caused by the distortion and the noise on the RX_DATA line, the duration of the baud rate measurement has been extended.

Previously, as described above, this determination was based on the measurement of the START bit duration. Now, this measurement is based on the duration of START bit + bit0. Bit0 is the first bit following the START bit. In fact, the counter which is started at the falling edge of START bit is no longer stopped at next rising edge (end of START bit), but it is stopped at the next falling edge (end of bit0). As the character sent is always a "A" (41h) or a "a" (61h), this second falling edge will always be present and it will indicate the end of bit0. Once this counter is stopped, the result is divided by 2 and used by the BRM to determine the incoming baud rate.

NOTE

UBRC register contains the result of this division by two, in consequence it reflects the measurement of the duration of one bit.

64.4.5.6.2.1 New Autobaud Counter Stopped bit and Interrupt

A new bit has been added in USR2 register: ACST (USR2[11]). This bit is set immediately after the determination of the baud rate.

So,

- if ADNIMP is not set (default), ACST is set to 1 after the end of bit0,
- If ADNIMP is set to 1, ACST is set to 1 at the end of START bit.

If ACIEN (UCR3[0]) is set to 1, ACST will flag an interrupt on *interrupt_uart* signal. This interrupt informs the ARM platform that the BRM has just been set with the result of the bit length measurement. If needed, the ARM platform can perform a read of UBMR (or UBRC) register and determine by itself the baud rate measured. Then the ARM platform has the possibility to correct the BRM registers with the nearest standardized baud rate.

NOTE

ACST is set only if ADBR is set to 1, for example, the UART is autobauding.

Clear the ACST bit by writing 1 to it. Writing 0 to the ACST bit has no effect.

64.4.6 Escape Sequence Detection

An escape sequence typically consists of 3 characters entered in rapid succession (such as +++). Because these are valid characters by themselves, the time between characters determines if it is a valid escape sequence.

Too much time between two of the "+" characters is interpreted as two "+" characters, and not part of an escape sequence.

The software chooses the escape character and writes its value to the UART Escape Character Register (UESC). The software must also enable escape detection feature by setting ESCEN (UCR2[11]) to 1. The hardware compares this value to incoming characters in the RxFIFO. When an escape character is detected, the internal escape timer starts to count. The software specifies a time-out value for the maximum allowable time between 2 successive escape characters (see the table below). The escape timer is programmable in intervals of 2 ms to a maximum interval of 8.192 seconds.

Table 64-13. Escape Timer Scaling

UTIM Register	Maximum Time Between Specified Escape Characters
0x000	2 ms
0x001	4 ms
0x002	6 ms
0x003	8 ms
0x004	10 ms
...	...
0F8	498 ms
0F9	500 ms
...	...
9C3	5 s
...	...
FFD	8.188 s
FFE	8.190 s
FFF	8.192 s
NOTE: To calculate the time interval: (UTIM_Value + 1) x 0.002 = Time_Interval Example: (09C3 + 1) x 0.002 = 5 s.	

The escape sequence detection feature is available for all the reference frequencies. Before using Escape Sequence Detection, the user must fill the ONEMS register. This 24-bit register must contain the value of the UART internal frequency divided by 1000. The internal frequency is obtained after the UART internal divider which is applied on *module_clock* clock.

Example I:

- If the input clock *module_clock* frequency is 66.5 MHz.
- And if the input clock *module_clock* is divided by 2 with the internal divider: UFCR[9:7] = 3'b100

$$ONEMS = \frac{66.5 \times 10^6}{2 \times 1000} = 33250 = 81E2h$$

Figure 64-8. Calculation of Frequency for ONEMS Register

Example II:

- If the input clock *module_clock* frequency is 66.5 MHz.
- And if the input clock *module_clock* is divided by 1with the internal divider: UFCR[9:7] = 3'b101

$$ONEMS = \frac{66.5 \times 10^6}{1000} = 66500 = 103C4h$$

Figure 64-9. Calculation of Frequency for ONEMS Register

The escape sequence detection interrupt is asserted when the escape sequence interrupt enable (ESCI) bit is set and an escape sequence is detected (ESCF set). Clear the ESCF bit by writing 1 to it. Writing 0 to the ESCF bit has no effect.

64.5 Binary Rate Multiplier (BRM)

The BRM sub-block receives *ref_clk* (*module_clock* clock after divider). From this clock, and with integer and non-integer division, BRM generates a 16x baud rate clock .

The UART transmitte will shift data out based on this 16x baud rate clock. The UART receiver will sample the serial data line based on this 16x baud rate clock. The input and output frequency ratio is programmed in the UART BRM Incremental Register (UBIR) and UART BRM MOD Register (UBMR). The output frequency is divided by the input frequency to produce this ratio. For integer division, set the UBIR = 0x000F and write the divisor to the UBMR register. All values written to these registers must be one less than the actual value to eliminate division by 0 (undefined), and to increase the maximum range of the registers.

Updating the BRM registers requires writing to both registers. The UBIR register must be written before writing to the UBMR register. If only one register is written to by the software, the BRM continues to use the previous values.

The following examples show how to determine what values are to be programmed into UBIR and UBMR for a given reference frequency and desired baud rate. The following equation can be used to help determine these values:

$$\text{BaudRate} = \frac{\text{Ref Freq}}{\left(16 \times \frac{\text{UBMR} + 1}{\text{UBIR} + 1} \right)}$$

Figure 64-10. Frequency and Baud Rate for UBIR and UBMR

With:

Reference Frequency (Hz): UART Reference Frequency (*module_clock* after RFDIV divider)

Baud Rate (bit/s): Desired baud rate.

Integer Division ÷ 21

Reference Frequency = 19.44 MHz

UBIR = 0x000F

UBMR = 0x0014

Baud Rate = 925.7 kbit/s

NOTE

Observe that each value written to the registers is one less than the actual value.

Non-Integer Division

```
Reference Frequency = 16 MHz
Desired Baud Rate = 920 Kbits/s
```

$$\frac{UBMR + 1}{UBIR + 1} = \frac{RefFreq}{16 \times BaudRate} = \frac{16 \times 10^6}{16 \times 920 \times 10^3} = 1.087$$

```
Ratio = 1.087 = 1087 / 1000
UBIR = 999 (decimal)= 0x3E7
UBMR = 1086 (decimal)= 0x43E
Non-Integer Division
Reference Frequency = 25 MHz
Desired Baud Rate = 920 kbit/s
Ratio = 1.69837 = 625 / 368
UBIR = 367 (decimal)= 0x16F
UBMR = 624 (decimal)= 0x270
```

Non-Integer Division

```
Reference Frequency: 30 MHz
Desired Baud Rate = 115.2 kbit/s
Ratio = 16.276043 = 65153 / 4003
UBIR = 4002 (decimal) = 0x0FA2
UBMR = 65152 (decimal) = 0xFE80
```

64.6 Infrared Interface

64.6.1 Generalities-Infrared

The Infrared interface is selected when IREN (UCR1[7]) is set to 1.

The Infrared Interface is compatible with IrDA Serial Infrared Physical Layer Specification. In this specification, a "zero" is represented by a positive pulse, and a "one" is represented by no pulse (line remains low).

In the UART:

In TX: For each "zero" to be transmitted, a narrow positive pulse which is 3/16 of a bit time is generated. For each "one" to be transmitted no pulse is generated (output is low). External circuitry has to be provided to drive an Infrared LED.

In RX: When receiving, a narrow negative pulse is expected for each "zero" transmitted while no pulse is expected for each "one" transmitted (input is high).

NOTE

Rx part of IR block expects to receive an inverted signal compared to IrDA specification. Circuitry external to the IC transforms the Infrared signal to an electrical signal.

The IR interface has an edge triggered interrupt (IRINT). This interrupt validates a zero bit being received. This interrupt is enabled by writing a "one" to ENIRI bit.

The behavior of Infrared Interface is determined by 3 bits INVT (UCR3[1]), INVR (UCR4[9]) and IRSC (UCR4[5]).

64.6.2 Inverted Transmission and Reception bits (INVT & INVR)

The values of INVT and INVR depend of the IrDA transceiver connected on the TXD_IR and RXD_IR pins of the UART. If this transceiver is not inverting on both paths Tx and Rx, a Zero is represented by a positive pulse and a One is represented by no pulse (line remains low). In this case, the bit INVT must be set to 0 and the bit INVR must be set to 1 (because Rx IR block expects an inverted signal).

On the contrary user must set INVT=1 and INVR=0 if both paths of the transceiver are inverting, that is, a Zero is represented as a negative pulse and a One is represented by no pulse (line remains high). The transceiver can also be inverting on only one path (Tx or Rx), in this case INVT and INVR must be together equal to 1 or to 0, depending on which path is inverted.

64.6.3 InfraRed Special Case (IRSC) Bit

The value to apply to IRSC bit is based on 2 parameters: the baud rate and the Minimum Pulse Duration (MPD) of the transceiver.

According to IrDA Standard Specification, for SIR (Serial IR) baud rates from 2.4 Kbit/s to 115.2 Kbit/s this nominal pulse duration is equal to 3/16 of a bit duration (at the selected baud rate). But, for all the baud rates a Minimum Pulse Duration is also specified. According to IrDA Standard, a Zero is represented by a light pulse, so the IrDA transceiver can't emit a light pulse shorter than the MPD. For SIR, the MPD is constant and equal to 1.41 µs.

But user must take into account the electrical MPD associated with the transceiver on the receiver path. Typically this value is 2.0 µs, but for some manufacturers MPD can go down to 1.0 µs.

In order to understand the meaning of IRSC bit, one must understand how the RX path works in IrDA mode.

When the UART is in IrDA mode, a Zero is not only detected by the state of the RXD_IR line, but also with the duration of the pulse. This pulse duration can be measured with 2 different clocks. In this case, clock is selected with the IRSC bit.

- If IRSC = 0, the clock used is the BRM clock.
- If IRSC = 1, the clock used is the UART internal clock (UART clock after the divider (RFDIV)).

In normal operation, IRSC=0. This means that at any time, the user must ensure that the frequency of BRM_clock is high enough to measure the pulse. The pulse must last at least 2 BRM clock cycles. If this condition is not fulfilled, IRSC must be set to 1.

Let's examine two examples, for a Minimum Pulse Duration equal to the MPD from the IrDA SIR specification (i.e., 1.41 µs).

1: Calculation of BRM Clock Period (Clock Period < 1.41 µs)

The user wants to receive IrDA data at 115.2 Kbit/s. The UBIR and UBMR registers are set in order to create the BRM_clock with a frequency of 16*baud rate = 16 * 115.2K = 1.843 MHz. But at the same time, in order to correctly detect the pulse, the user must be sure that 2* BRM_clock period is lower than 1.41 µs. Lets check:

BRM_clock period = 1/1843000 = 542 ns

So 2*BRM_clock period = 1.09 µs < 1.41 µs. It is fine.

2: Calculation of BRM Clock Period (Clock Period > 1.41 µs)

This time the user wants to receive at 19.2 Kbit/s. So, the BRM_clock is set to 16*19200 = 307.2 kHz. Let's check if 2* BRM_clock period < 1.41 µs:

1. BRM_clock period =1/307200 = 3.25 µs

   So 2*BRM_clock period = 6.50 µs >> 1.41 µs. It doesn't work.

   So, in this case, the BRM clock can't be used to measure the pulse duration and the user must select the UART internal clock by setting IRSC =1.

NOTE

Like for Escape character detection, when IR Special Case is enabled (IRSC=1), the UART must measure a duration. In order to do that, the user must fill the ONEMS register. Refer to Escape Sequence Detection.

64.6.4 IrDA interrupt

Serial infrared mode (SIR) uses an edge triggered interrupt flag IRINT (USR2[8]). When INVR =0, detection of a falling edge on the RXD pin asserts the IRINT bit. When INVR=1, detection of a rising edge on the RXD pin asserts the IRINT bit. When IRINT and ENIRI bits are both asserted, the *interrupt_uart* interrupt is asserted. Clear the IRINT bit by writing 1 to it. Writing 0 to the IRINT bit has no effect.

64.6.5 Conclusion about IrDA

Before using the UART in IrDA, the baud rate limit must be calculated. This baud rate limit will inform the user if IRSC bit has to be set or not.

Let's determine this limit:

As already described, if IRSC = 0, the following condition must always be fulfilled

$$2 \times BRMClockPeriod < MinPulseDuration$$

Figure 64-11. Calculation of Baud Rate

So,

$$BRMClockFrequency > \frac{2}{MPD}$$

So, knowing BRM_clock frequency = 16 * Baud Rate, we get:

$$BaudRate > \frac{1}{8 \times MinPulseDuration}$$

So, the user needs to set IRSC = 0 when:

- If Minimum Pulse Duration = 2.5 us and Baud Rate > 50 Kbit/s.
- If Minimum Pulse Duration = 2.0 us and Baud Rate > 62.5 Kbit/s.
- If Minimum Pulse Duration = 1.41 us and Baud Rate > 88.6 Kbit/s.

NOTE

For baud rates lower than the limit, IRSC must be set to1.

64.6.6 Programming IrDA Interface

64.6.6.1 High Speed

As an example, the following sequence can be used to program the IrDA interface in order to send and receive characters at 115.2 Kbit/s.

Assumptions:

- Input UART clock = 90 MHz
- Internal clock divider = 3 (divide Input UART clock by 3)
- Baud rate = 115.2 Kbit/s
- IrDA transceiver is not inverting on both channels: for Tx and Rx, a Zero is represented by a positive pulse, and a One is represented by no pulse (line stays low).
- Interrupt: Sent to ARM platform when 1 char is received into the Rx FIFO (RDR)

Registers values and Programming orders:

```
UCR1 = 0x0085
UCR1[7] = IREN = 1: Enable IR interface
UCR1[0] = UARTEN = 1: Enable UART
UTS = 0x0000
UFCR = 0x0981
TXTL[5:0] = 0x02: Default value
RFDIV[2:0] = 0x3: Divide Input UART clock by 3 (resulting internal clock  is 30 MHz)
RXTL[5:0] = 0x01: Default value
UBIR = 0x0202
UBMR = 0x20BE Baud rate = 115.2 kbit/s with internal clock = 30 MHz
UCR2 = 0x4027
UCR2[14] = IRTS = 1: Ignore level of RTS input signal
UCR2[5] = WS = 1: Characters are 8-bit length
UCR2[2] = TXEN = 1: Enable Rx path
UCR2 [1] = RXEN = 1: Enable Tx path
UCR2[0] = SRST_B = 1: No software reset
UCR3 = 0x0000

         UCR4 = 0x8201
CTSTL[5:0] = 0x20: Default value
UCR4[9] = INVR = 1: Inverted Infrared Reception (because IrDA transceiver is not inverting)
UCR4[1] = DREN = 1: To enable RDR interrupt (sent when one char is received)
```

The UART is ready to send a character as soon as there is a write into UTXD register. And an interrupt will be sent to ARM platform when a character is received.

64.6.6.2 Low Speed

This time, we keep the same assumptions but the speed is now 9.6 Kbit/s. So, this baud rate is below the limit (even with a Min. Pulse Duration of 2.5 us) and thus IRSC must be set to 1.

Assumptions:

- Input UART clock = 90 MHz
- Internal clock divider = 3 (divide Input UART clock by 3)
- Baud rate = 9.6 Kbit/s
- IrDA transceiver is not inverting on both channels: for Tx and Rx, a Zero is represented by a positive pulse, and a One is represented by no pulse (line stays low).
- Interrupt: Sent to ARM platform when 1 char is received into the Rx FIFO (RDR).

Registers values and Programming orders:

```
UCR1 = 0x0085
UCR1[7] = IREN = 1: Enable IR interface
UCR1[0] = UARTEN = 1: Enable UART
UFCR = 0x0981
UFCR[15:10] = TXTL[5:0] = 0x02: Default value
RFDIV[2:0] = 0x3: Divide Input UART clock by 3 (resulting internal clock  is 30 MHz)
UFCR[5:0] = RXTL[5:0] = 0x01: Default value
UBIR = 0x00FF
UBMR = 0xC354 Baud rate = 9.6 kbit/s with internal clock = 30 MHz
UCR2 = 0x4027
UCR2[14] = IRTS = 1: Ignore level of RTS input signal
UCR2[5] = WS = 1: Characters are 8-bit length
UCR2[2] = TXEN = 1: Enable Rx path
UCR2 [1] = RXEN = 1: Enable Tx path
UCR2[0] = SRST_B = 1: No software reset
UCR3 = 0x0000
UCR3[1] = INVT = 0: Positive pulse represents 0.
UCR4 = 0x8221
UCR4[15:10] = CTSTL[5:0] = 0x20: Default value
UCR4[9] = INVR = 1: Inverted Infrared Reception (because IrDA transceiver is not inverting)
UCR4[5] = IRSC = 1: Because data rate is below the limit and thus the UART internal clock is
used to measure the pulse duration.
UCR4[1] = DREN = 1: To enable RDR interrupt (sent when one char is received)
```

The UART is now ready to send a character as soon as there is a write into UTXD register. An interrupt will be sent to ARM platform when a character is received.

64.7 9-bit RS-485 Mode

64.7.1 Generalities

The UART provides a 9-bit mode to facilitate multidrop (RS-485) network communication. To enable this mode, set MDEN bit in the UMCR register to 1. When 9-bit RS-485 mode is enabled, UART transmitter can transmit the ninth bit (9^{th} bit) set by TXB8, and UART receiver can differentiate between data frames (9^{th} bit = 0) and address frames (9^{th} bit = 1).

The CTS_B pin can be used to control RS-485 output driver outside the chip.

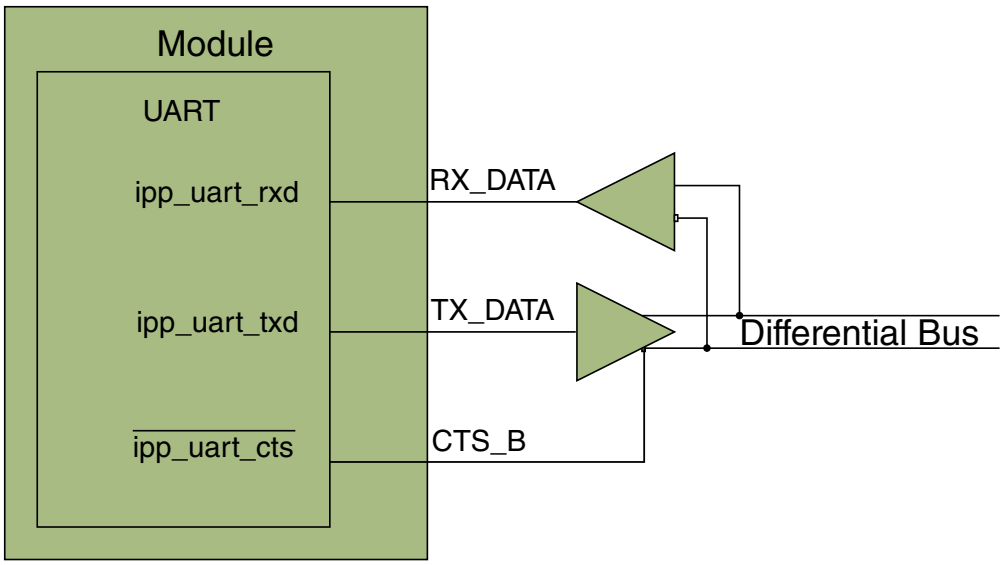


Figure 64-12. RS-485 driver connection (UART in DCE mode)

64.7.2 Transmit 9-bit RS-485 frames

To transmit 9-bit RS-485 frames, user need to enable parity (PREN=1) to enable trasmitting the ninth data bit, set 8-bit data word size (WS=1), and write TXB8 (UMCR[2]) as the 9th bit (bit [8]) to be transmitted (write '0' to TXB8 to transmit a data frame, write '1' to transmit a address frame). The other data bit [7:0] is written to TxFIFO by writing to the UTXD same as normal RS-232 operation.

64.7.3 Receive 9-bit RS-485 frames

To receive 9-bit RS-485 frames, user need to enable parity (PREN=1) to enable receiving the ninth data bit, set 8-bit data word size (WS=1). The receiver will save the 9-bit data to RxFIFO, and user should read the 9th databit (bit [8]) by reading the PRERR (URXD[10]) bit, and read data bit [7:0] by reading the RX_DATA (URXD[7:0]).

There are two slave address detect modes, normal detect mode and automatic detect mode, and can be selected by SLAM (UMCR[1]).

64.7.3.1 RS-485 Slave Address Normal Detect Mode

To enable Normal Detect mode, clear SLAM (UMCR[1] to 0). The receiver ignores all data frames (9^{th} bit = 0) until an address frame is received (9^{th} bit = 1). At that time, the slave address detected (SAD = USR1[3]) bit is asserted and the *interrupt_uart* interrupt is generated (if SADEN = UMCR[3] = 1). The address byte and sebsequent bytes are all put into RxFIFO along with their 9^{th} bit. The UART will also generate DMA request *dma_req_rx* when the RxFIFO reaches the selected threshold (controlled by RXTL) if receive ready DMA (RXDMAEN = UCR1[8]) request is enabled.

User should read the 9^{th} databit (bit [8]) by reading the PRERR (URXD[10]) bit, and read data bit [7:0] by reading the RX_DATA (URXD[7:0]).

In this mode, once the UART has detected a 9^{th} bit is equal to '1', it will always save the subsequent frames to RxFIFO. So the software must decide whether the address and data in RxFIFO are needed or not.

64.7.3.2 RS-485 Slave Address Automatic Detect Mode

To enable Automatic Detect Mode, set SLAM (UMCR[1]) to 1. The receiver tries to detect an address byte (frame 9^{th} bit = 1) that matches the programmed SLADDR (UMCR[15:8]) character. If the received byte is a data or an address byte that does not match the programmed SLADDR character, the receiver will discard these data.

Once the UART receives a matching address byte, it will assert the slave address detected (SAD = USR1[3]) bit and the *interrupt_uart* interrupt will be generated (if SADEN = UMCR[3] = 1). The address byte and sebsequent bytes are all put into RxFIFO along with their 9^{th} bit. If receive ready DMA(RXDMAEN = UCR1[8]) request is enabled, the UART will also generate DMA request *dma_req_rx* when the RxFIFO reaches the selected threshold (controlled by RXTL).

If another address byte is received and this address byte does not match SLADDR character, the receiver will discard the address byte and subsequent data byte. If the address byte again matches SLADDR character, the receiver will put this address byte and subsequent data byte in the RxFIFO along with their 9^{th} bit.

User should read the 9^{th} databit (bit [8]) by reading the PRERR (URXD[10]) bit, and read data bit [7:0] by reading the RX_DATA (URXD[7:0]).

See Initialization for 9-bit RS-485 programming guide.

64.8 Low Power Modes

These modes are controlled by the signals *doze_req* and *stop_req*. The control/status/data registers won't change when getting in/out of low power modes.

Table 64-14. UART Low Power State Operation

	Normal State (*doze_req* = 1'b0 & *stop_req* = 1'b0)	Doze State (*doze_req* = 1'b1)		Stop State (*stop_req* = 1'b1)
		DOZE bit = 0	DOZE bit = 1	
UART-Clock	ON	ON	ON	OFF
UART Serial / IrDA	ON	ON	OFF	OFF

64.8.1 UART Operation in System Doze Mode

While in Doze State (when *doze_req* input pin is set to 1'b1), the UART behavior depends on the DOZE (UCR1[1]) control bit.

While the DOZE bit is negated, the UART serial interface is enabled. While the system is in the Doze State, and the DOZE bit is asserted, the UART is disabled. If the Doze State is entered with the DOZE bit asserted while the UART serial interface was receiving or transmitting data, it will complete the receive/transmit of the current character and signal to the far-end transmitter/receiver to stop sending/receiving.

64.8.2 UART Operation in System Stop Mode

The internal baud rate clocks of the transmitter and receiver are gated off if the *stop_req* signal to UART is asserted. Even though the clocks at the input of the UART continue to run during system Stop mode, the UART will not do any transmission or reception.

The following UART interrupts wake the ARM platform processor from STOP mode:

- RTS (RTSD)
- IrDA Asynchronous WAKE (AIRINT)
- Asynchronous WAKE (AWAKE)
- RI (RIDELT in DTE mode only)
- DCD (DCDDELT in DTE mode only)
- DTR (DTRD in DCE mode only)
- DSR (DTRD in DTE mode only)

When an asynchronous WAKE (awake) interrupt exits the ARM platform from STOP mode, make sure that a dummy character is sent first because the first character may not be received correctly.

64.8.3 Power Saving Method in UART

The RXEN (UCR2[1]), TXEN (UCR2[2]) and UARTEN (UCR1[0]) bits are set by the user and provide software control of low-power modes.

Setting the UARTEN (UCR1[0]) bit to 0 shuts off the receiver and transmitter logic and the associated clocks.

If the UART is used only in transmit mode, UARTEN and TXEN must be set to 1. If the UART is used only in receive mode, UARTEN and RXEN must be set to 1. Setting TXEN or RXEN to 0 allows to save a lot of power.

64.9 UART Operation in System Debug State

The bit UTS [11] controls whether the UART will respond to the input signal *debug_req*, or whether it will continue to run as normal.

If the UART is programmed to respond to *debug_req*:

1. The UART will halt all operations upon detecting the *debug_req* input.
2. A transfer in progress, either to/from a core (using the IP Bus interface) or to/from an external device, will be completed before halting. This means a single byte/word transfer, not an entire FIFO. Reception of any further data from an external device will be disabled.
3. Internal registers will continue to be writable and readable using the IP Bus interface. A read will leave the contents unaffected.
4. The RX FIFO is affected in debug mode in the following way:
   - All writes into the RX FIFO are prevented.
   - The bit RXDBG (UTS[9]) is used to select the readability of the RX FIFO during debug mode:

     RXDBG = 0: hold the read pointer at the location it had upon entering debug mode, and URXD register returns only the data value at that location, no matter how many reads attempted.

RXDBG = 1, selectable at any time: Allow to read the characters received in Rx FIFO. It will not be possible to re-read previously read locations, nor will it be possible to readjust the read pointer to the value it had prior to entering debug mode.

64.10 Reset

This section describes how to reset the block and explains special requirements related to reset.

64.10.1 Hardware reset

All of registers, FIFOs, state machines and sequential elements can be reset to their initial values by hardware reset or power on reset.

64.10.2 Software reset

The status registers USR1 and USR2, BRM registers UBIR and UBMR, TxFIFO and RxFIFO, and transmitter and receiver state machines can be reset by software reset. Internal logic will keep the software reset asserted for about 4 *module_clock* cycles.

Programmer can follow the following software reset sequence:

1. Clear the SRST_B bit (UCR2[0])
2. Wait for software reset complete: poll SOFTRST bit (UTS[0]) until it is 0.
3. Re-program baud rate registers: Re-write UBIR and UBMR.

64.11 Transfer Error

The UART can generate a transfer error on the peripheral bus in the following cases:

- Core is writing into a read-only register.
- Core is accessing (read or write) an unused location within the assigned address space reserved to UART.

- Core is writing into UTXD register with transmit interface disabled (TXEN=0 or UARTEN=0)
- Core is reading URXD register with receive interface disabled (RXEN=0 or UARTEN=0)

64.12 Functional Timing

This section includes timing diagrams for functional signaling.

64.12.1 IrDA Mode

According to IrDA specification, the low speed (115.2Kbit/s and below) IR frame format is compatible with UART frame.

In this figure, an example data 0x65 is used.

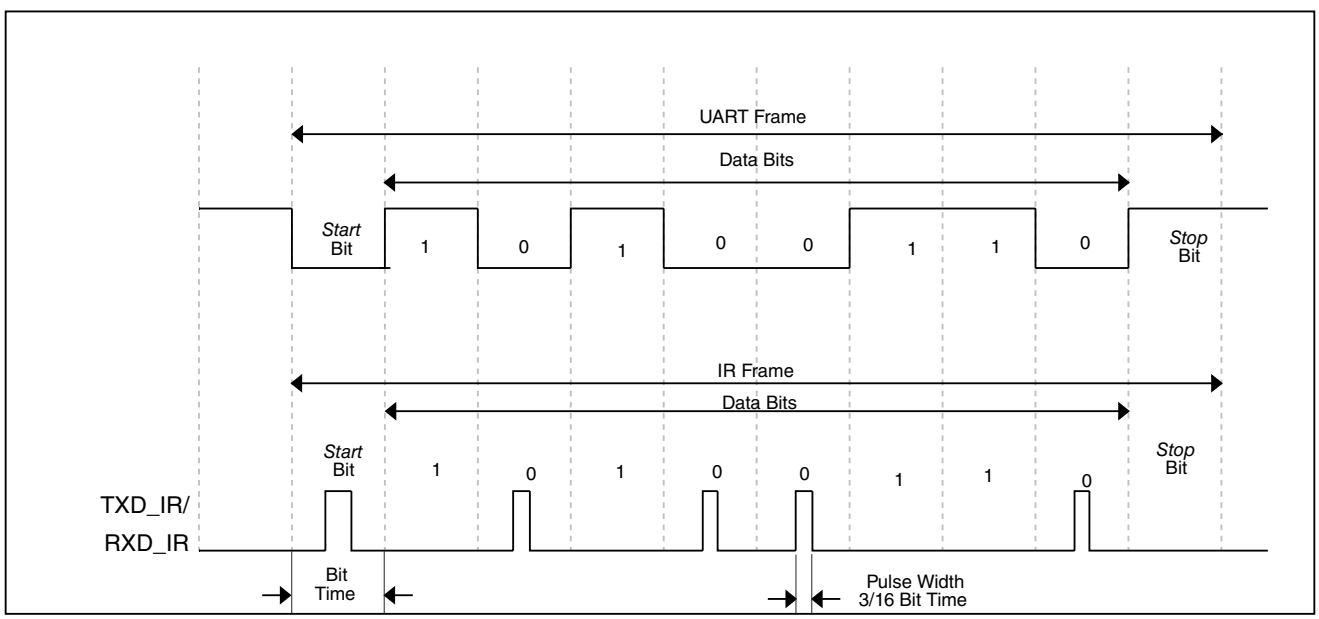


Figure 64-13. Timing diagram of Low Speed IR (<=115.2 Kbit/s) Data Line

64.13 Initialization

64.13.1 Programming the UART in RS-232 mode

As an example, the following sequence can be used to program the UART in order to send and receive characters in RS-232 mode.

Assumptions:

- Input uart clock = 100 MHz
- Baud rate = 921.6Kbps
- Data bits = 8 bits
- Parity = Even
- Stop bits = 1 bit
- Flow control = Hardware

Main program:

1. UCR1 = 0x0001

   Enable the UART.

2. UCR2 = 0x2127

   Set hardware flow control, data format and enable transmitter and receiver.

3. UCR3 = 0x0704

   Set UCR3[RXDMUXSEL] = 1.

4. UCR4 = 0x7C00

   Set CTS trigger level to 31,

5. UFCR = 0x089E

   Set internal clock divider = 5 (divide input uart clock by 5). So the reference clock is 100MHz/5 = 20MHz.

   Set TXTL = 2 and RXTL = 30.

6. UBIR = 0x08FF
7. UBMR = 0x0C34

   In the above two steps, set baud rate to 921.6Kbps based on the 20MHz reference clock.

8. UCR1 = 0x2201

   Enable the TRDY and RRDY interrupts.

9. UMCR = 0x0000

UMCR stay at default value 0x0000

Interrupt service routine for the transmitter:

- Write characters into UTXD

The TRDY interrupt will be automatically de-asserted when the data level of the TxFIFO exceeds the TXTL=2. Note: For the first time the interrupt may be de-asserted after 4 characters are written into the TxFIFO because of the shift register.

Interrupt service routine for the receiver:

- Read characters from URXD

The RRDY interrupt will be automatically de-asserted when the data level of the RxFIFO is below the RXTL=30.

64.13.2 Programming the UART in 9-bit RS-485 mode

As an example, the following sequence can be used to program the UART in order to send and receive frames in RS-485 mode.

Assumptions:

- Input uart clock = 100 MHz
- Baud rate = 5Mbps

Main program:

1. UCR1 = 0x0001

   Enable the UART.

2. UCR2 = 0x4127

   Set software flow control ($\overline{\text{CTS}}$ pin is controlled by UCR2[12]), enable parity(enable 9^{th} bit rxd/txd), 8-bit word size , and enable transmitter and receiver.

3. UCR4 = 0x7C00

   Set CTS trigger level to 31,

4. UFCR = 0x0A9E

   Set RFDIV = 5 (divide input uart clock by 1), so the reference clock is 100MHz. Set UART in DCE mode (RS-485 driver connection outside the chip is the same as Figure 64-12)

Set TXTL = 2 and RXTL = 30.

5. UBIR = 0x0003
6. UBMR = 0x0004

   In the above two steps, set baud rate to 5Mbps based on the 100MHz reference clock.

7. UCR1 = 0x2001 when UART as a master ,

   or UCR1 = 0x0201 (or 0x0101) when UART as a slave.

   Enable TRDY interrupt when UART as a master, enable RRDY interrupt or DMA request when UART as a slave.

8. UMCR = 0xA50B

   Enable 9-bit RS-485 mode, enable SAD interrupt, set automatic slave address detect mode, set slave address is 0xA5.

Interrupt service routine for the transmitter:

- Transmit data: write its ninth bit (bit[8]) to UMCR[2], write its bit [7:0] into UTXD[7:0]

The TRDY interrupt will be automatically de-asserted when the data level of the TxFIFO exceeds the TXTL=2.

Note: For the first time the interrupt may be de-asserted after 4 characters are written into the TxFIFO because of the shift register.

Interrupt service routine for the receiver:

- Receive data: read its ninth bit (bit[8]) from URXD[10] , read its bit [7:0] from URXD[7:0].

Note: in RS-485 mode, URXD[10] bit is not the parity error, instead it holds the ninth bit (bit[8]) of the received data.

The SAD interrupt can not de-assert automatically, it needs MCU write 1 to USR1[3] to clear it . The RRDY interrupt or DMA request will be automatically de-asserted when the data level of the RxFIFO is below the RXTL=30.

64.14 References

- EIA/TIA-232-F Interface Standard

http://www.eia.org, http://www.tiaonline.org/standards

- IrDA Standard

  http://www.irda.org

64.15 UART Memory Map/Register Definition

UART supports 8-bit, 16-bit and 32-bit accesses to 32-bit memory-mapped addresses. Any access to unmapped memory location will yield a transfer error.

All registers except the ONEMS described in this section are 16-bit registers. The ONEMS register is a 24-bit register.

- For 32-bit write accesses, the upper two bytes will not be taken into account.
- For 32-bit read accesses the upper two bytes will return 0.

The ONEMS register is expanded from 16 bits to 24 bits in order to support the high frequency of the BRM internal clock *ref_clk* (*module_clock* after divider). The ONEMS register can be accessed as 8 bits, 16 bits or 32 bits.

- For 32-bit write accesses, the most significant byte of the ONEMS will be discarded.
- For 32-bit read accesses, the most significant byte of the ONEMS will be read as 0.

UART memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
202_0000	UART Receiver Register (UART1_URXD)	32	R	0000_0000h	64.15.1/ 5220
202_0040	UART Transmitter Register (UART1_UTXD)	32	W	0000_0000h	64.15.2/ 5222
202_0080	UART Control Register 1 (UART1_UCR1)	32	R/W	0000_0000h	64.15.3/ 5223
202_0084	UART Control Register 2 (UART1_UCR2)	32	R/W	0000_0001h	64.15.4/ 5225
202_0088	UART Control Register 3 (UART1_UCR3)	32	R/W	0000_0700h	64.15.5/ 5228
202_008C	UART Control Register 4 (UART1_UCR4)	32	R/W	0000_8000h	64.15.6/ 5230
202_0090	UART FIFO Control Register (UART1_UFCR)	32	R/W	0000_0801h	64.15.7/ 5232

Table continues on the next page...

UART memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
202_0094	UART Status Register 1 (UART1_USR1)	32	R/W	0000_2040h	64.15.8/ 5234
202_0098	UART Status Register 2 (UART1_USR2)	32	R/W	0000_4028h	64.15.9/ 5237
202_009C	UART Escape Character Register (UART1_UESC)	32	R/W	0000_002Bh	64.15.10/ 5239
202_00A0	UART Escape Timer Register (UART1_UTIM)	32	R/W	0000_0000h	64.15.11/ 5240
202_00A4	UART BRM Incremental Register (UART1_UBIR)	32	R/W	0000_0000h	64.15.12/ 5240
202_00A8	UART BRM Modulator Register (UART1_UBMR)	32	R/W	0000_0000h	64.15.13/ 5241
202_00AC	UART Baud Rate Count Register (UART1_UBRC)	32	R	0000_0004h	64.15.14/ 5241
202_00B0	UART One Millisecond Register (UART1_ONEMS)	32	R/W	0000_0000h	64.15.15/ 5242
202_00B4	UART Test Register (UART1_UTS)	32	R/W	0000_0060h	64.15.16/ 5243
202_00B8	UART RS-485 Mode Control Register (UART1_UMCR)	32	R/W	0000_0000h	64.15.17/ 5244
21E_8000	UART Receiver Register (UART2_URXD)	32	R	0000_0000h	64.15.1/ 5220
21E_8040	UART Transmitter Register (UART2_UTXD)	32	W	0000_0000h	64.15.2/ 5222
21E_8080	UART Control Register 1 (UART2_UCR1)	32	R/W	0000_0000h	64.15.3/ 5223
21E_8084	UART Control Register 2 (UART2_UCR2)	32	R/W	0000_0001h	64.15.4/ 5225
21E_8088	UART Control Register 3 (UART2_UCR3)	32	R/W	0000_0700h	64.15.5/ 5228
21E_808C	UART Control Register 4 (UART2_UCR4)	32	R/W	0000_8000h	64.15.6/ 5230
21E_8090	UART FIFO Control Register (UART2_UFCR)	32	R/W	0000_0801h	64.15.7/ 5232
21E_8094	UART Status Register 1 (UART2_USR1)	32	R/W	0000_2040h	64.15.8/ 5234
21E_8098	UART Status Register 2 (UART2_USR2)	32	R/W	0000_4028h	64.15.9/ 5237
21E_809C	UART Escape Character Register (UART2_UESC)	32	R/W	0000_002Bh	64.15.10/ 5239
21E_80A0	UART Escape Timer Register (UART2_UTIM)	32	R/W	0000_0000h	64.15.11/ 5240
21E_80A4	UART BRM Incremental Register (UART2_UBIR)	32	R/W	0000_0000h	64.15.12/ 5240

Table continues on the next page...

UART memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21E_80A8	UART BRM Modulator Register (UART2_UBMR)	32	R/W	0000_0000h	64.15.13/ 5241
21E_80AC	UART Baud Rate Count Register (UART2_UBRC)	32	R	0000_0004h	64.15.14/ 5241
21E_80B0	UART One Millisecond Register (UART2_ONEMS)	32	R/W	0000_0000h	64.15.15/ 5242
21E_80B4	UART Test Register (UART2_UTS)	32	R/W	0000_0060h	64.15.16/ 5243
21E_80B8	UART RS-485 Mode Control Register (UART2_UMCR)	32	R/W	0000_0000h	64.15.17/ 5244
21E_C000	UART Receiver Register (UART3_URXD)	32	R	0000_0000h	64.15.1/ 5220
21E_C040	UART Transmitter Register (UART3_UTXD)	32	W	0000_0000h	64.15.2/ 5222
21E_C080	UART Control Register 1 (UART3_UCR1)	32	R/W	0000_0000h	64.15.3/ 5223
21E_C084	UART Control Register 2 (UART3_UCR2)	32	R/W	0000_0001h	64.15.4/ 5225
21E_C088	UART Control Register 3 (UART3_UCR3)	32	R/W	0000_0700h	64.15.5/ 5228
21E_C08C	UART Control Register 4 (UART3_UCR4)	32	R/W	0000_8000h	64.15.6/ 5230
21E_C090	UART FIFO Control Register (UART3_UFCR)	32	R/W	0000_0801h	64.15.7/ 5232
21E_C094	UART Status Register 1 (UART3_USR1)	32	R/W	0000_2040h	64.15.8/ 5234
21E_C098	UART Status Register 2 (UART3_USR2)	32	R/W	0000_4028h	64.15.9/ 5237
21E_C09C	UART Escape Character Register (UART3_UESC)	32	R/W	0000_002Bh	64.15.10/ 5239
21E_C0A0	UART Escape Timer Register (UART3_UTIM)	32	R/W	0000_0000h	64.15.11/ 5240
21E_C0A4	UART BRM Incremental Register (UART3_UBIR)	32	R/W	0000_0000h	64.15.12/ 5240
21E_C0A8	UART BRM Modulator Register (UART3_UBMR)	32	R/W	0000_0000h	64.15.13/ 5241
21E_C0AC	UART Baud Rate Count Register (UART3_UBRC)	32	R	0000_0004h	64.15.14/ 5241
21E_C0B0	UART One Millisecond Register (UART3_ONEMS)	32	R/W	0000_0000h	64.15.15/ 5242
21E_C0B4	UART Test Register (UART3_UTS)	32	R/W	0000_0060h	64.15.16/ 5243
21E_C0B8	UART RS-485 Mode Control Register (UART3_UMCR)	32	R/W	0000_0000h	64.15.17/ 5244

UART memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21F_0000	UART Receiver Register (UART4_URXD)	32	R	0000_0000h	64.15.1/ 5220
21F_0040	UART Transmitter Register (UART4_UTXD)	32	W	0000_0000h	64.15.2/ 5222
21F_0080	UART Control Register 1 (UART4_UCR1)	32	R/W	0000_0000h	64.15.3/ 5223
21F_0084	UART Control Register 2 (UART4_UCR2)	32	R/W	0000_0001h	64.15.4/ 5225
21F_0088	UART Control Register 3 (UART4_UCR3)	32	R/W	0000_0700h	64.15.5/ 5228
21F_008C	UART Control Register 4 (UART4_UCR4)	32	R/W	0000_8000h	64.15.6/ 5230
21F_0090	UART FIFO Control Register (UART4_UFCR)	32	R/W	0000_0801h	64.15.7/ 5232
21F_0094	UART Status Register 1 (UART4_USR1)	32	R/W	0000_2040h	64.15.8/ 5234
21F_0098	UART Status Register 2 (UART4_USR2)	32	R/W	0000_4028h	64.15.9/ 5237
21F_009C	UART Escape Character Register (UART4_UESC)	32	R/W	0000_002Bh	64.15.10/ 5239
21F_00A0	UART Escape Timer Register (UART4_UTIM)	32	R/W	0000_0000h	64.15.11/ 5240
21F_00A4	UART BRM Incremental Register (UART4_UBIR)	32	R/W	0000_0000h	64.15.12/ 5240
21F_00A8	UART BRM Modulator Register (UART4_UBMR)	32	R/W	0000_0000h	64.15.13/ 5241
21F_00AC	UART Baud Rate Count Register (UART4_UBRC)	32	R	0000_0004h	64.15.14/ 5241
21F_00B0	UART One Millisecond Register (UART4_ONEMS)	32	R/W	0000_0000h	64.15.15/ 5242
21F_00B4	UART Test Register (UART4_UTS)	32	R/W	0000_0060h	64.15.16/ 5243
21F_00B8	UART RS-485 Mode Control Register (UART4_UMCR)	32	R/W	0000_0000h	64.15.17/ 5244
21F_4000	UART Receiver Register (UART5_URXD)	32	R	0000_0000h	64.15.1/ 5220
21F_4040	UART Transmitter Register (UART5_UTXD)	32	W	0000_0000h	64.15.2/ 5222
21F_4080	UART Control Register 1 (UART5_UCR1)	32	R/W	0000_0000h	64.15.3/ 5223
21F_4084	UART Control Register 2 (UART5_UCR2)	32	R/W	0000_0001h	64.15.4/ 5225
21F_4088	UART Control Register 3 (UART5_UCR3)	32	R/W	0000_0700h	64.15.5/ 5228

Table continues on the next page...

UART memory map (continued)

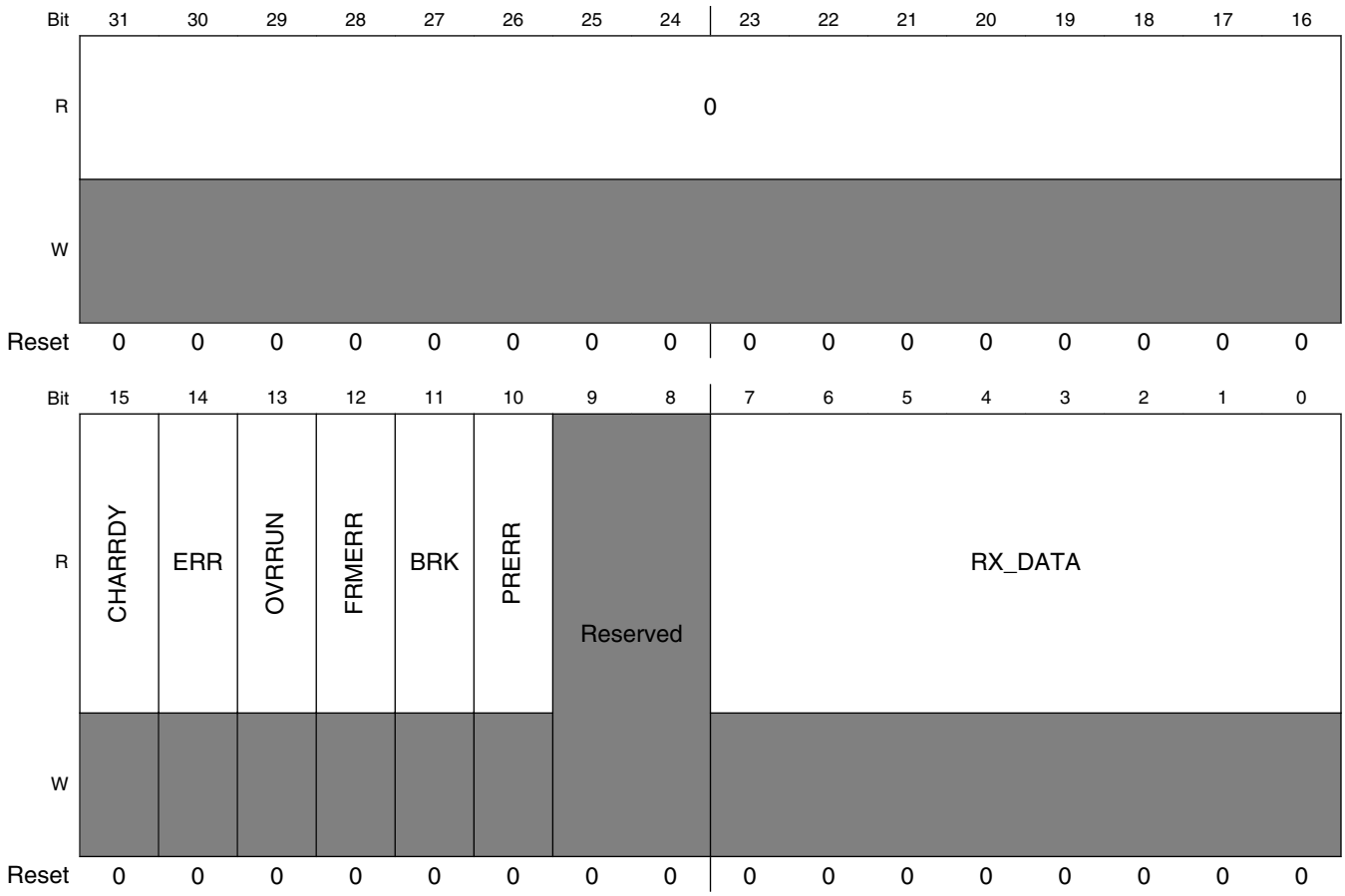
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21F_408C	UART Control Register 4 (UART5_UCR4)	32	R/W	0000_8000h	64.15.6/ 5230
21F_4090	UART FIFO Control Register (UART5_UFCR)	32	R/W	0000_0801h	64.15.7/ 5232
21F_4094	UART Status Register 1 (UART5_USR1)	32	R/W	0000_2040h	64.15.8/ 5234
21F_4098	UART Status Register 2 (UART5_USR2)	32	R/W	0000_4028h	64.15.9/ 5237
21F_409C	UART Escape Character Register (UART5_UESC)	32	R/W	0000_002Bh	64.15.10/ 5239
21F_40A0	UART Escape Timer Register (UART5_UTIM)	32	R/W	0000_0000h	64.15.11/ 5240
21F_40A4	UART BRM Incremental Register (UART5_UBIR)	32	R/W	0000_0000h	64.15.12/ 5240
21F_40A8	UART BRM Modulator Register (UART5_UBMR)	32	R/W	0000_0000h	64.15.13/ 5241
21F_40AC	UART Baud Rate Count Register (UART5_UBRC)	32	R	0000_0004h	64.15.14/ 5241
21F_40B0	UART One Millisecond Register (UART5_ONEMS)	32	R/W	0000_0000h	64.15.15/ 5242
21F_40B4	UART Test Register (UART5_UTS)	32	R/W	0000_0060h	64.15.16/ 5243
21F_40B8	UART RS-485 Mode Control Register (UART5_UMCR)	32	R/W	0000_0000h	64.15.17/ 5244

64.15.1 UART Receiver Register (UARTx_URXD)

NOTE

The UART will yield a transfer error on the peripheral bus when core is reading URXD register with receive interface disabled (RXEN=0 or UARTEN=0).

Address: Base address + 0h offset



UARTx_URXD field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15 CHARRDY	Character Ready. This read-only bit indicates an invalid read when the FIFO becomes empty and software tries to read the same old data. This bit should not be used for polling for data written to the RX FIFO. 0 Character in RX_DATA field and associated flags are invalid. 1 Character in RX_DATA field and associated flags valid and ready for reading.

Table continues on the next page...

UARTx_URXD field descriptions (continued)

Field	Description
14 ERR	**Error Detect.** Indicates whether the character present in the RX_DATA field has an error (OVRRUN, FRMERR, BRK or PRERR) status. The ERR bit is updated and valid for each received character. 0 No error status was detected 1 An error status was detected
13 OVRRUN	**Receiver Overrun.** This read-only bit, when HIGH, indicates that the corresponding character was stored in the last position (32nd) of the Rx FIFO. Even if a 33rd character has not been detected, this bit will be set to '1' for the 32nd character. 0 No RxFIFO overrun was detected 1 A RxFIFO overrun was detected
12 FRMERR	**Frame Error.** Indicates whether the current character had a framing error (a missing stop bit) and is possibly corrupted. FRMERR is updated for each character read from the RxFIFO. 0 The current character has no framing error 1 The current character has a framing error
11 BRK	**BREAK Detect.** Indicates whether the current character was detected as a BREAK character. The data bits and the stop bit are all 0. The FRMERR bit is set when BRK is set. When odd parity is selected, PRERR is also set when BRK is set. BRK is valid for each character read from the RxFIFO. 0 The current character is not a BREAK character 1 The current character is a BREAK character
10 PRERR	**In RS-485 mode, it holds the ninth data bit (bit [8]) of received 9-bit RS-485 data** **In RS232/IrDA mode, it is the Parity Error flag**. Indicates whether the current character was detected with a parity error and is possibly corrupted. PRERR is updated for each character read from the RxFIFO. When parity is disabled, PRERR always reads as 0. 0 = No parity error was detected for data in the RX_DATA field 1 = A parity error was detected for data in the RX_DATA field
9–8 -	This field is reserved. **Reserved**
RX_DATA	**Received Data**. Holds the received character. In 7-bit mode, the most significant bit (MSB) is forced to 0. In 8-bit mode, all bits are active.

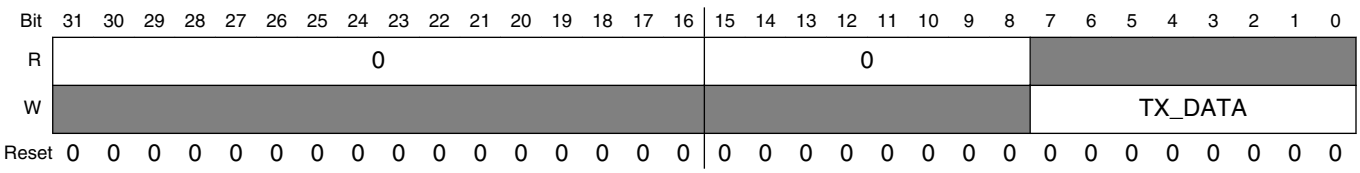
64.15.2 UART Transmitter Register (UARTx_UTXD)

NOTE

The UART will yield a transfer error on the peripheral bus when core is writing into UART_URXD register with transmit interface disabled (TXEN=0 or UARTEN=0).

Memory space between UART_URXD and UART_UTXD registers is reserved. Any read or write access to this space will be considered as an invalid access and yield a transfer error.

Address: Base address + 40h offset



UARTx_UTXD field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15–8 Reserved	This read-only field is reserved and always has the value 0.
TX_DATA	**Transmit Data**. Holds the parallel transmit data inputs. In 7-bit mode, D7 is ignored. In 8-bit mode, all bits are used. Data is transmitted least significant bit (LSB) first. A new character is transmitted when the TX_DATA field is written. The TX_DATA field must be written only when the TRDY bit is high to ensure that corrupted data is not sent.

64.15.3 UART Control Register 1 (UARTx_UCR1)

Address: Base address + 80h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADEN	ADBR	TRDYEN	IDEN	ICD		RRDYEN	RXDMAEN	IREN	TXMPTYEN	RTSDEN	SNDBRK	TXDMAEN	ATDMAEN	DOZE	UARTEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

UARTx_UCR1 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15 ADEN	**Automatic Baud Rate Detection Interrupt Enable**. Enables/Disables the automatic baud rate detect complete (ADET) bit to generate an interrupt (*interrupt_uart* = 0). 0 Disable the automatic baud rate detection interrupt 1 Enable the automatic baud rate detection interrupt
14 ADBR	**Automatic Detection of Baud Rate**. Enables/Disables automatic baud rate detection. When the ADBR bit is set and the ADET bit is cleared, the receiver detects the incoming baud rate automatically. The ADET flag is set when the receiver verifies that the incoming baud rate is detected properly by detecting an ASCII character "A" or "a" (0x41 or 0x61). 0 Disable automatic detection of baud rate 1 Enable automatic detection of baud rate
13 TRDYEN	**Transmitter Ready Interrupt Enable**. Enables/Disables the transmitter Ready Interrupt (TRDY) when the transmitter has one or more slots available in the TxFIFO. The fill level in the TXFIFO at which an interrupt is generated is controlled by TxTL bits. When TRDYEN is negated, the transmitter ready interrupt is disabled. **NOTE:** An interrupt will be issued as long as TRDYEN and TRDY are high even if the transmitter is not enabled. In general, user should enable the transmitter before enabling the TRDY interrupt. 0 Disable the transmitter ready interrupt 1 Enable the transmitter ready interrupt
12 IDEN	**Idle Condition Detected Interrupt Enable**. Enables/Disables the IDLE bit to generate an interrupt (*interrupt_uart* = 0). 0 Disable the IDLE interrupt 1 Enable the IDLE interrupt

Table continues on the next page...

UARTx_UCR1 field descriptions (continued)

Field	Description
11–10 ICD	**Idle Condition Detect**. Controls the number of frames RXD is allowed to be idle before an idle condition is reported. 00 Idle for more than 4 frames 01 Idle for more than 8 frames 10 Idle for more than 16 frames 11 Idle for more than 32 frames
9 RRDYEN	**Receiver Ready Interrupt Enable**. Enables/Disables the RRDY interrupt when the RxFIFO contains data. The fill level in the RxFIFO at which an interrupt is generated is controlled by the RXTL bits. When RRDYEN is negated, the receiver ready interrupt is disabled. 0 Disables the RRDY interrupt 1 Enables the RRDY interrupt
8 RXDMAEN	**Receive Ready DMA Enable**. Enables/Disables the receive DMA request *dma_req_rx* when the receiver has data in the RxFIFO. The fill level in the RxFIFO at which a DMA request is generated is controlled by the RXTL bits. When negated, the receive DMA request is disabled. 0 Disable DMA request 1 Enable DMA request
7 IREN	**Infrared Interface Enable**. Enables/Disables the IR interface. See the IR interface description in Infrared Interface, for more information. Note: MDEN(UMCR[0]) must be cleared to 0 when using IrDA interface. See Table 64-1 0 Disable the IR interface 1 Enable the IR interface
6 TXMPTYEN	**Transmitter Empty Interrupt Enable**. Enables/Disables the transmitter FIFO empty (TXFE) interrupt. *interrupt_uart*. When negated, the TXFE interrupt is disabled. **NOTE:** An interrupt will be issued as long as TXMPTYEN and TXFE are high even if the transmitter is not enabled. In general, user should enable the transmitter before enabling the TXFE interrupt. 0 Disable the transmitter FIFO empty interrupt 1 Enable the transmitter FIFO empty interrupt
5 RTSDEN	**RTS Delta Interrupt Enable**. Enables/Disables the RTSD interrupt. The current status of the RTS_B pin is read in the RTSS bit. 0 Disable RTSD interrupt 1 Enable RTSD interrupt
4 SNDBRK	**Send BREAK**. Forces the transmitter to send a BREAK character. The transmitter finishes sending the character in progress (if any) and sends BREAK characters until SNDBRK is reset. Because the transmitter samples SNDBRK after every bit is transmitted, it is important that SNDBRK is asserted high for a sufficient period of time to generate a valid BREAK. After the BREAK transmission completes, the UART transmits 2 mark bits. The user can continue to fill the TxFIFO and any characters remaining are transmitted when the BREAK is terminated. 0 Do not send a BREAK character 1 Send a BREAK character (continuous 0s)
3 TXDMAEN	**Transmitter Ready DMA Enable**. Enables/Disables the transmit DMA request *dma_req_tx* when the transmitter has one or more slots available in the TxFIFO. The fill level in the TxFIFO that generates the *dma_req_tx* is controlled by the TXTL bits.

Table continues on the next page...

UARTx_UCR1 field descriptions (continued)

Field	Description
	NOTE: A DMA request will be issued as long as TXDMAEN and TRDY are high even if the transmitter is not enabled. In general, user should enable the transmitter before enabling the transmit DMA request. 0 Disable transmit DMA request 1 Enable transmit DMA request
2 ATDMAEN	**Aging DMA Timer Enable**. Enables/Disables the receive DMA request *dma_req_rx* for the aging timer interrupt (triggered with AGTIM flag in USR1[8]). 0 Disable AGTIM DMA request 1 Enable AGTIM DMA request
1 DOZE	**DOZE**. Determines the UART enable condition in the DOZE state. When *doze_req* input pin is at '1', (the ARM Platform executes a doze instruction and the system is placed in the Doze State), the DOZE bit affects operation of the UART. While in the Doze State, if this bit is asserted, the UART is disabled. See the description in Low Power Modes. 0 The UART is enabled when in DOZE state 1 The UART is disabled when in DOZE state
0 UARTEN	**UART Enable**. Enables/Disables the UART. If UARTEN is negated in the middle of a transmission, the transmitter stops and pulls the TXD line to a logic 1. UARTEN must be set to 1 before any access to UTXD and URXD registers, otherwise a transfer error is returned. This bit can be set to 1 along with other bits in this register. There is no restriction to the sequence of programing this bit and other control registers. 0 Disable the UART 1 Enable the UART

64.15.4 UART Control Register 2 (UARTx_UCR2)

Address: Base address + 84h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ESCI	IRTS	CTSC	CTS	ESCEN	RTEC		PREN	PROE	STPB	WS	RTSEN	ATEN	TXEN	RXEN	SRST
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

UARTx_UCR2 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15 ESCI	**Escape Sequence Interrupt Enable**. Enables/Disables the ESCF bit to generate an interrupt. 0 Disable the escape sequence interrupt 1 Enable the escape sequence interrupt
14 IRTS	**Ignore RTS Pin**. Forces the RTS input signal presented to the transmitter to always be asserted (set to low), effectively ignoring the external pin. When in this mode, the RTS pin serves as a general purpose input. 0 Transmit only when the RTS pin is asserted 1 Ignore the RTS pin
13 CTSC	CTS **Pin Control**. Controls the operation of the CTS_B module output. When CTSC is asserted, the CTS_B module output is controlled by the receiver. When the RxFIFO is filled to the level of the programmed trigger level and the start bit of the overflowing character (TRIGGER LEVEL + 1) is validated, the CTS_B module output is negated to indicate to the far-end transmitter to stop transmitting. When the trigger level is programmed for less than 32, the receiver continues to receive data until the RxFIFO is full. When the CTSC bit is negated, the CTS_B module output is controlled by the CTS bit. On reset, because CTSC is cleared to 0, the CTS_B pin is controlled by the CTS bit, which again is cleared to 0 on reset. This means that on reset the CTS_B signal is negated. 0 The CTS_B pin is controlled by the CTS bit 1 The CTS_B pin is controlled by the receiver
12 CTS	**Clear to Send**. Controls the CTS_B pin when the CTSC bit is negated. CTS has no function when CTSC is asserted. 0 The CTS_B pin is high (inactive) 1 The CTS_B pin is low (active)
11 ESCEN	**Escape Enable**. Enables/Disables the escape sequence detection logic. 0 Disable escape sequence detection 1 Enable escape sequence detection
10–9 RTEC	**Request to Send Edge Control**. Selects the edge that triggers the RTS interrupt. This has no effect on the RTS delta interrupt. RTEC has an effect only when RTSEN = 1 (see Table 64-8). 00 Trigger interrupt on a rising edge 01 Trigger interrupt on a falling edge 1X Trigger interrupt on any edge
8 PREN	**Parity Enable**. Enables/Disables the parity generator in the transmitter and parity checker in the receiver. When PREN is asserted, the parity generator and checker are enabled, and disabled when PREN is negated. 0 Disable parity generator and checker 1 Enable parity generator and checker
7 PROE	**Parity Odd/Even**. Controls the sense of the parity generator and checker. When PROE is high, odd parity is generated and expected. When PROE is low, even parity is generated and expected. PROE has no function if PREN is low. 0 Even parity 1 Odd parity

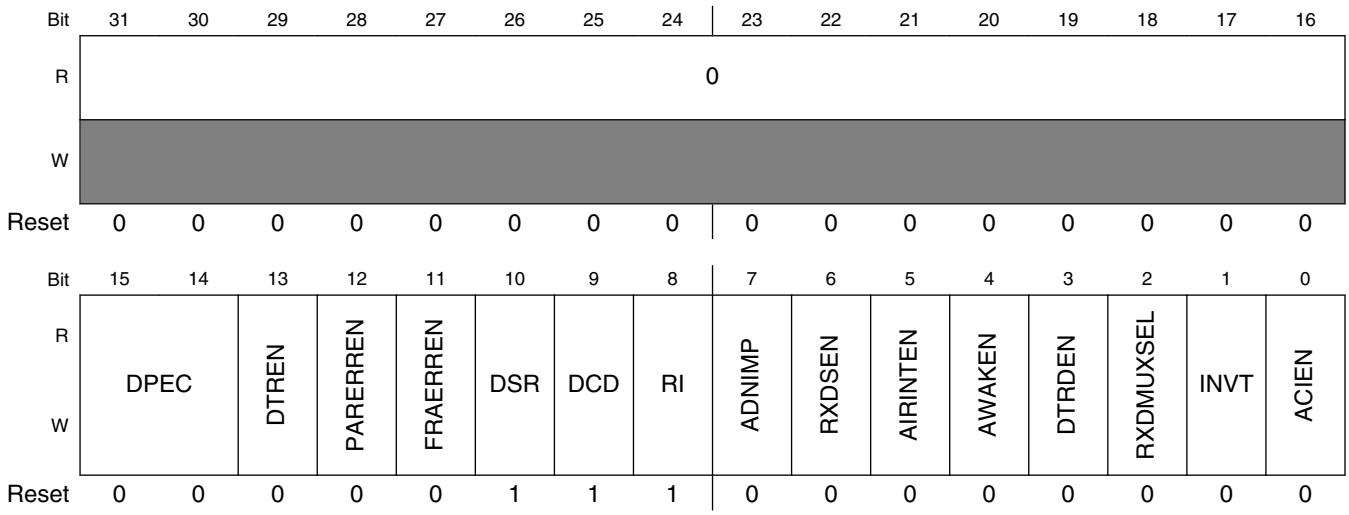
Table continues on the next page...

UARTx_UCR2 field descriptions (continued)

Field	Description
6 STPB	**Stop**. Controls the number of stop bits after a character. When STPB is low, 1 stop bit is sent. When STPB is high, 2 stop bits are sent. STPB also affects the receiver. 0 The transmitter sends 1 stop bit. The receiver expects 1 or more stop bits. 1 The transmitter sends 2 stop bits. The receiver expects 2 or more stop bits.
5 WS	**Word Size**. Controls the character length. When WS is high, the transmitter and receiver are in 8-bit mode. When WS is low, they are in 7-bit mode. The transmitter ignores bit 7 and the receiver sets bit 7 to 0. WS can be changed in-between transmission (reception) of characters, however not when a transmission (reception) is in progress, in which case the length of the current character being transmitted (received) is unpredictable. 0 7-bit transmit and receive character length (not including START, STOP or PARITY bits) 1 8-bit transmit and receive character length (not including START, STOP or PARITY bits)
4 RTSEN	**Request to Send Interrupt Enable.** Controls the RTS edge sensitive interrupt. When RTSEN is asserted and the programmed edge is detected on the RTS_B pin (the RTSF bit is asserted), an interrupt will be generated on the *interrupt_uart* pin. (See Table 64-8.) 0 Disable request to send interrupt 1 Enable request to send interrupt
3 ATEN	**Aging Timer Enable**. This bit is used to enable the aging timer interrupt (triggered with AGTIM) 0 AGTIM interrupt disabled 1 AGTIM interrupt enabled
2 TXEN	**Transmitter Enable**. Enables/Disables the transmitter. When TXEN is negated the transmitter is disabled and idle. When the UARTEN and TXEN bits are set the transmitter is enabled. If TXEN is negated in the middle of a transmission, the UART disables the transmitter immediately, and starts marking 1s. The transmitter FIFO cannot be written when this bit is cleared. 0 Disable the transmitter 1 Enable the transmitter
1 RXEN	**Receiver Enable**. Enables/Disables the receiver. When the receiver is enabled, if the RXD input is already low, the receiver does not recognize BREAK characters, because it requires a valid 1-to-0 transition before it can accept any character. 0 Disable the receiver 1 Enable the receiver
0 SRST	**Software Reset**. Once the software writes 0 to SRST_B, the software reset remains active for 4 *module_clock* cycles before the hardware deasserts SRST_B. The software can only write 0 to SRST_B. Writing 1 to SRST_B is ignored. 0 Reset the transmit and receive state machines, all FIFOs and register USR1, USR2, UBIR, UBMR, UBRC , URXD, UTXD and UTS[6-3]. 1 No reset

64.15.5 UART Control Register 3 (UARTx_UCR3)

Address: Base address + 88h offset



UARTx_UCR3 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15–14 DPEC	**DTR/DSR Interrupt Edge Control**. These bits control the edge of DTR_B (DCE) or DSR_B (DTE) on which an interrupt will be generated. An interrupt will only be generated if the DTREN bit is set. 00 interrupt generated on rising edge 01 interrupt generated on falling edge 1X interrupt generated on either edge
13 DTREN	**Data Terminal Ready Interrupt Enable**. When this bit is set, it will enable the status bit DTRF (USR2 [13]) (DTR/DSR edge sensitive interrupt) to cause an interrupt. 0 Data Terminal Ready Interrupt Disabled 1 Data Terminal Ready Interrupt Enabled
12 PARERREN	**Parity Error Interrupt Enable. Enables/Disables the interrupt. When asserted, PARERREN causes the PARITYERR bit to generate an interrupt.** 0 Disable the parity error interrupt 1 Enable the parity error interrupt
11 FRAERREN	**Frame Error Interrupt Enable. Enables/Disables the interrupt. When asserted, FRAERREN causes the FRAMERR bit to generate an interrupt.** 0 Disable the frame error interrupt 1 Enable the frame error interrupt
10 DSR	**Data Set Ready**. This bit is used by software to control the DSR/DTR module output for the modem interface. In DCE mode it applies to DSR_B and in DTE mode it applies to DTR_B. 0 DSR/ DTR pin is logic zero 1 DSR/ DTR pin is logic one

Table continues on the next page...

UARTx_UCR3 field descriptions (continued)

Field	Description
9 DCD	**Data Carrier Detect**. In DCE mode this bit is used by software to control the DCD_B module output for the modem interface. In DTE mode, when this bit is set, it will enable the status bit DCDDELT (USR2 (6)) to cause an interrupt. 0 DCD_B pin is logic zero (DCE mode) 1 DCD_B pin is logic one (DCE mode) 0 DCDDELT interrupt disabled (DTE mode) 1 DCDDELT interrupt enabled (DTE mode)
8 RI	**Ring Indicator**. In DCE mode this bit is used by software to control the RI_B module output for the modem interface. In DTE mode, when this bit is set, it will enable the status bit RIDELT (USR2 (10)) to cause an interrupt. 0 RI_B pin is logic zero (DCE mode) 1 RI_B pin is logic one (DCE mode) 0 RIDELT interrupt disabled (DTE mode) 1 RIDELT interrupt enabled (DTE mode)
7 ADNIMP	**Autobaud Detection Not Improved-. Disables new features of autobaud detection (See** Baud Rate Automatic Detection Protocol**, for more details).** 0 Autobaud detection new features selected 1 Keep old autobaud detection mechanism
6 RXDSEN	Receive Status Interrupt Enable. Controls the receive status interrupt (*interrupt_uart*). When this bit is enabled and RXDS status bit is set, the interrupt *interrupt_uart* will be generated. 0 Disable the RXDS interrupt 1 Enable the RXDS interrupt
5 AIRINTEN	Asynchronous IR WAKE Interrupt Enable. Controls the asynchronous IR WAKE interrupt. An interrupt is generated when AIRINTEN is asserted and a pulse is detected on the RXD pin. 0 Disable the AIRINT interrupt 1 Enable the AIRINT interrupt
4 AWAKEN	Asynchronous WAKE Interrupt Enable. Controls the asynchronous WAKE interrupt. An interrupt is generated when AWAKEN is asserted and a falling edge is detected on the RXD pin. 0 Disable the AWAKE interrupt 1 Enable the AWAKE interrupt
3 DTRDEN	**Data Terminal Ready Delta Enable**. Enables / Disables the asynchronous DTRD interrupt. When DTRDEN is asserted and an edge (rising or falling) is detected on DTR_B (in DCE mode) or on DSR_B (in DTE mode), then an interrupt is generated. 0 Disable DTRD interrupt 1 Enable DTRD interrupt
2 RXDMUXSEL	RXD Muxed Input Selected. Selects proper input pins for serial and Infrared input signal. **NOTE:** In this chip, UARTs are used in MUXED mode, so that this bit should always be set.
1 INVT	Invert TXD output in RS-232/RS-485 mode, set TXD active level in IrDA mode. In RS232/RS-485 mode(UMCR[0] = 1), if this bit is set to 1, the TXD output is inverted before transmitted. In **IrDA mode**, when INVT is cleared, the infrared logic block transmits a positive IR 3/16 pulse for all 0s and 0s are transmitted for 1s. When INVT is set (INVT = 1), the infrared logic block transmits an active low or negative infrared 3/16 pulse for all 0s and 1s are transmitted for 1s.

Table continues on the next page...

UARTx_UCR3 field descriptions (continued)

Field	Description
	0 **TXD is not inverted** 1 **TXD is inverted** 0 **TXD** Active low transmission 1 **TXD** Active high transmission
0 ACIEN	**Autobaud Counter Interrupt Enable.** This bit is used to enable the autobaud counter stopped interrupt (triggered with ACST (USR2[11]). 0 ACST interrupt disabled 1 ACST interrupt enabled

64.15.6 UART Control Register 4 (UARTx_UCR4)

Address: Base address + 8Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CTSTL						INVR	ENIRI	WKEN	IDDMAEN	IRSC	LPBYP	TCEN	BKEN	OREN	DREN
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

UARTx_UCR4 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15–10 CTSTL	**CTS Trigger Level**. Controls the threshold at which the CTS_B pin is deasserted by the RxFIFO. After the trigger level is reached and the CTS_B pin is deasserted, the RxFIFO continues to receive data until it is full. The CTSTL bits are encoded as shown in the Settings column. Settings 0 to 32 are in use. All other settings are Reserved. 000000 0 characters received 000001 1 characters in the RxFIFO ... — ... — 100000 32 characters in the RxFIFO (maximum)
9 INVR	**Invert RXD input in RS-232/RS-485 Mode, d**etermine **RXD input** logic level being sampled in **In IrDA mode.**

Table continues on the next page...

UARTx_UCR4 field descriptions (continued)

Field	Description
	In RS232/RS-485 Mode(UMCR[0] = 1), if this bit is set to 1, the RXD input is inverted before sampled. **In IrDA mode**,when cleared, the infrared logic block expects an active low or negative IR 3/16 pulse for 0s and 1s are expected for 1s. When INVR is set (INVR 1), the infrared logic block expects an active high or positive IR 3/16 pulse for 0s and 0s are expected for 1s. 0 **RXD input is not inverted** 1 **RXD input is inverted** 0 **RXD** active low detection 1 **RXD** active high detection
8 ENIRI	**Serial Infrared Interrupt Enable**. Enables/Disables the serial infrared interrupt. 0 Serial infrared Interrupt disabled 1 Serial infrared Interrupt enabled
7 WKEN	**WAKE Interrupt Enable**. Enables/Disables the WAKE bit to generate an interrupt. The WAKE bit is set at the detection of a start bit by the receiver. 0 Disable the WAKE interrupt 1 Enable the WAKE interrupt
6 IDDMAEN	**DMA IDLE Condition Detected Interrupt Enable** Enables/Disables the receive DMA request *dma_req_rx* for the IDLE interrupt (triggered with IDLE flag in USR2[12]). 0 DMA IDLE interrupt disabled 1 DMA IDLE interrupt enabled
5 IRSC	**IR Special Case**. Selects the clock for the vote logic. When set, IRSC switches the vote logic clock from the sampling clock to the UART reference clock. The IR pulses are counted a predetermined amount of time depending on the reference frequency. See InfraRed Special Case (IRSC) Bit. 0 The vote logic uses the sampling clock (16x baud rate) for normal operation 1 The vote logic uses the UART reference clock
4 LPBYP	**Low Power Bypass**. Allows to bypass the low power new features in UART. To use during debug phase. 0 Low power features enabled 1 Low power features disabled
3 TCEN	**TransmitComplete Interrupt Enable**. Enables/Disables the TXDC bit to generate an interrupt (*interrupt_uart* = 0) **NOTE:** An interrupt will be issued as long as TCEN and TXDC are high even if the transmitter is not enabled. In general, user should enable the transmitter before enabling the TXDC interrupt. 0 Disable TXDC interrupt 1 Enable TXDC interrupt
2 BKEN	**BREAK Condition Detected Interrupt Enable**. Enables/Disables the BRCD bit to generate an interrupt. 0 Disable the BRCD interrupt 1 Enable the BRCD interrupt
1 OREN	**Receiver Overrun Interrupt Enable**. Enables/Disables the ORE bit to generate an interrupt. 0 Disable ORE interrupt 1 Enable ORE interrupt

Table continues on the next page...

UARTx_UCR4 field descriptions (continued)

Field	Description
0 DREN	**Receive Data Ready Interrupt Enable**. Enables/Disables the RDR bit to generate an interrupt. 0 Disable RDR interrupt 1 Enable RDR interrupt

64.15.7 UART FIFO Control Register (UARTx_UFCR)

Address: Base address + 90h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TXTL						RFDIV			DCEDTE	RXTL					
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1

UARTx_UFCR field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15–10 TXTL	**Transmitter Trigger Level**. Controls the threshold at which a maskable interrupt is generated by the TxFIFO. A maskable interrupt is generated whenever the data level in the TxFIFO falls below the selected threshold. The bits are encoded as shown in the Settings column. Settings 0 to 32 are in use. All other settings are Reserved. 000000 Reserved 000001 Reserved 000010 TxFIFO has 2 or fewer characters ... — ... — 011111 TxFIFO has 31 or fewer characters 100000 TxFIFO has 32 characters (maximum)
9–7 RFDIV	Reference Frequency Divider. Controls the divide ratio for the reference clock. The input clock is *module_clock*. The output from the divider is *ref_clk* which is used by BRM to create the 16x baud rate oversampling clock (*brm_clk*). 000 Divide input clock by 6 001 Divide input clock by 5

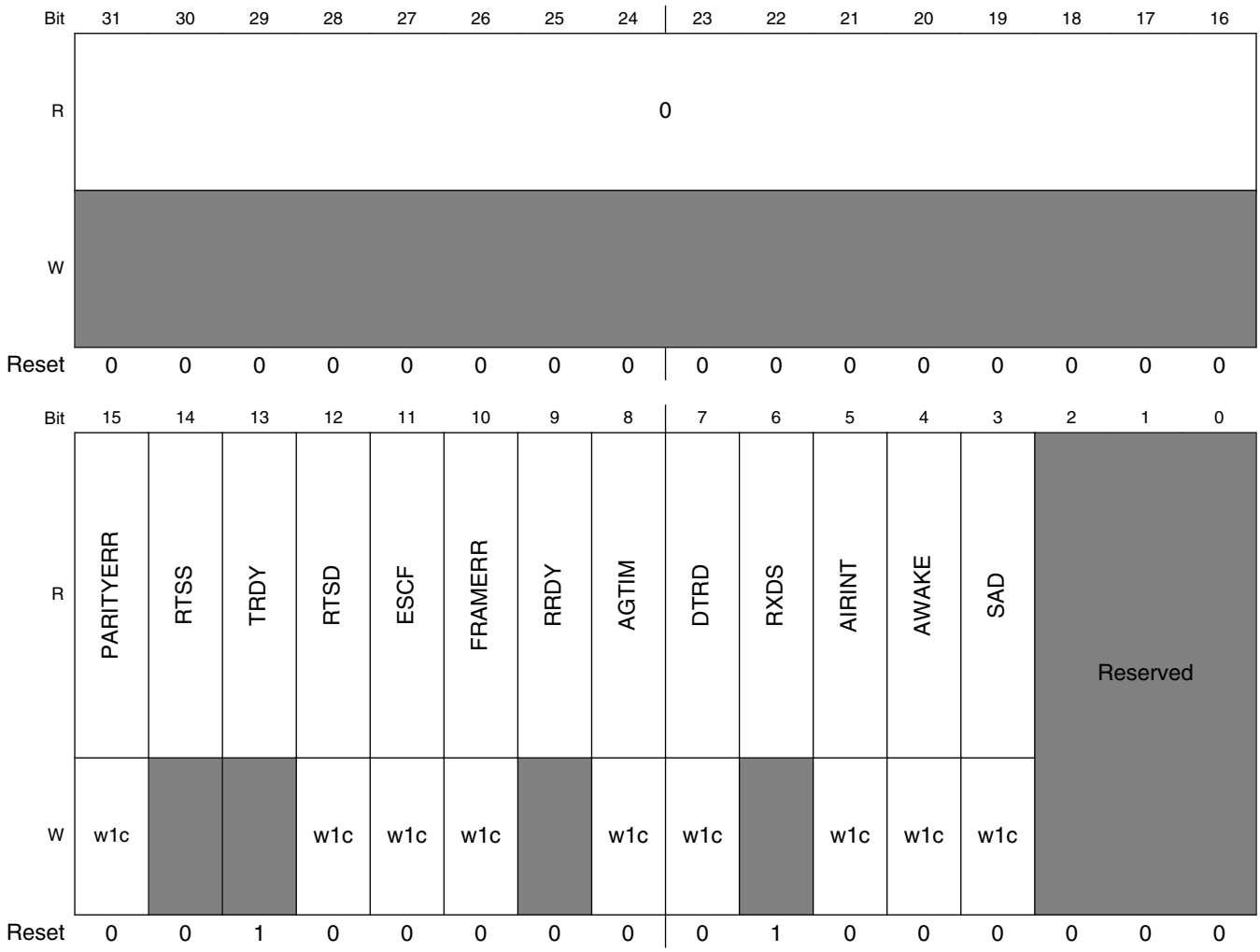
Table continues on the next page...

UARTx_UFCR field descriptions (continued)

Field	Description
	010 Divide input clock by 4 011 Divide input clock by 3 100 Divide input clock by 2 101 Divide input clock by 1 110 Divide input clock by 7 111 Reserved
6 DCEDTE	**DCE/DTE mode select**. Select UART as data communication equipment (DCE mode) or as data terminal equipment (DTE mode). 0 DCE mode selected 1 DTE mode selected
RXTL	**Receiver Trigger Level**. Controls the threshold at which a maskable interrupt is generated by the RxFIFO. A maskable interrupt is generated whenever the data level in the RxFIFO reaches the selected threshold. The RXTL bits are encoded as shown in the Settings column. Setting 0 to 32 are in use. All other settings are Reserved. 000000 0 characters received 000001 RxFIFO has 1 character ... — ... — 011111 RxFIFO has 31 characters 100000 RxFIFO has 32 characters (maximum)

64.15.8 UART Status Register 1 (UARTx_USR1)

Address: Base address + 94h offset



UARTx_USR1 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15 PARITYERR	**Parity Error Interrupt Flag**. Indicates a parity error is detected. PARITYERR is cleared by writing 1 to it. Writing 0 to PARITYERR has no effect. When parity is disabled, PARITYERR always reads 0. At reset, PARITYERR is set to 0. 0 No parity error detected 1 Parity error detected (write 1 to clear)
14 RTSS	RTS_B **Pin Status**. Indicates the current status of the RTS_B pin. A "snapshot" of RTS_B is taken immediately before RTSS is presented to the data bus. RTSS cannot be cleared because all writes to RTSS are ignored. At reset, RTSS is set to 0.

Table continues on the next page...

UARTx_USR1 field descriptions (continued)

Field	Description
	0 The RTS_B module input is high (inactive) 1 The RTS_B module input is low (active)
13 TRDY	**Transmitter Ready Interrupt / DMA Flag**. Indicates that the TxFIFO emptied below its target threshold and requires data. TRDY is automatically cleared when the data level in the TxFIFO exceeds the threshold set by TXTL bits. At reset, TRDY is set to 1. 0 The transmitter does not require data 1 The transmitter requires data (interrupt posted)
12 RTSD	**RTS Delta.** Indicates whether the RTS_B pin changed state. It (RTSD) generates a maskable interrupt. When in STOP mode, RTS assertion sets RTSD and can be used to wake the processor. The current state of the RTS_B pin is available on the RTSS bit. Clear RTSD by writing 1 to it. Writing 0 to RTSD has no effect. At reset, RTSD is set to 0. 0 RTS_B pin did not change state since last cleared 1 RTS_B pin changed state (write 1 to clear)
11 ESCF	**Escape Sequence Interrupt Flag**. Indicates if an escape sequence was detected. ESCF is asserted when the ESCEN bit is set and an escape sequence is detected in the RxFIFO. Clear ESCF by writing 1 to it. Writing 0 to ESCF has no effect. 0 No escape sequence detected 1 Escape sequence detected (write 1 to clear).
10 FRAMERR	**Frame Error Interrupt Flag**. Indicates that a frame error is detected. The *interrupt_uart* interrupt will be generated if a frame error is detected and the interrupt is enabled. Clear FRAMERR by writing 1 to it. Writing 0 to FRAMERR has no effect. 0 No frame error detected 1 Frame error detected (write 1 to clear)
9 RRDY	**Receiver Ready Interrupt / DMA Flag**. Indicates that the RxFIFO data level is above the threshold set by the RXTL bits. (See the RXTL bits description in UART FIFO Control Register (UART_UFCR) for setting the interrupt threshold.) When asserted, RRDY generates a maskable interrupt or DMA request. RRDY is automatically cleared when data level in the RxFIFO goes below the set threshold level. At reset, RRDY is set to 0. 0 No character ready 1 Character(s) ready (interrupt posted)
8 AGTIM	Ageing Timer Interrupt Flag. Indicates that data in the RxFIFO has been idle for a time of 8 character lengths (where a character length consists of 7 or 8 bits, depending on the setting of the WS bit in UCR2, with the bit time corresponding to the baud rate setting) and FIFO data level is less than RxFIFO threshold level (RXTL in the UFCR). Clear by writing a 1 to it. 0 AGTIM is not active 1 AGTIM is active (write 1 to clear)
7 DTRD	**DTR Delta.** Indicates whether DTR_B (in DCE mode) or DSR_B (in DTE mode) pins changed state. DTRD generates a maskable interrupt if DTRDEN (UCR3[3]) is set. Clear DTRD by writing 1 to it. Writing 0 to DTRD has no effect. 0 DTR_B (DCE) or DSR_B (DTE) pin did not change state since last cleared 1 DTR_B (DCE) or DSR_B (DTE) pin changed state (write 1 to clear)
6 RXDS	Receiver IDLE Interrupt Flag. Indicates that the receiver state machine is in an IDLE state, the next state is IDLE, and the receive pin is high. RXDS is automatically cleared when a character is received. RXDS is active only when the receiver is enabled.

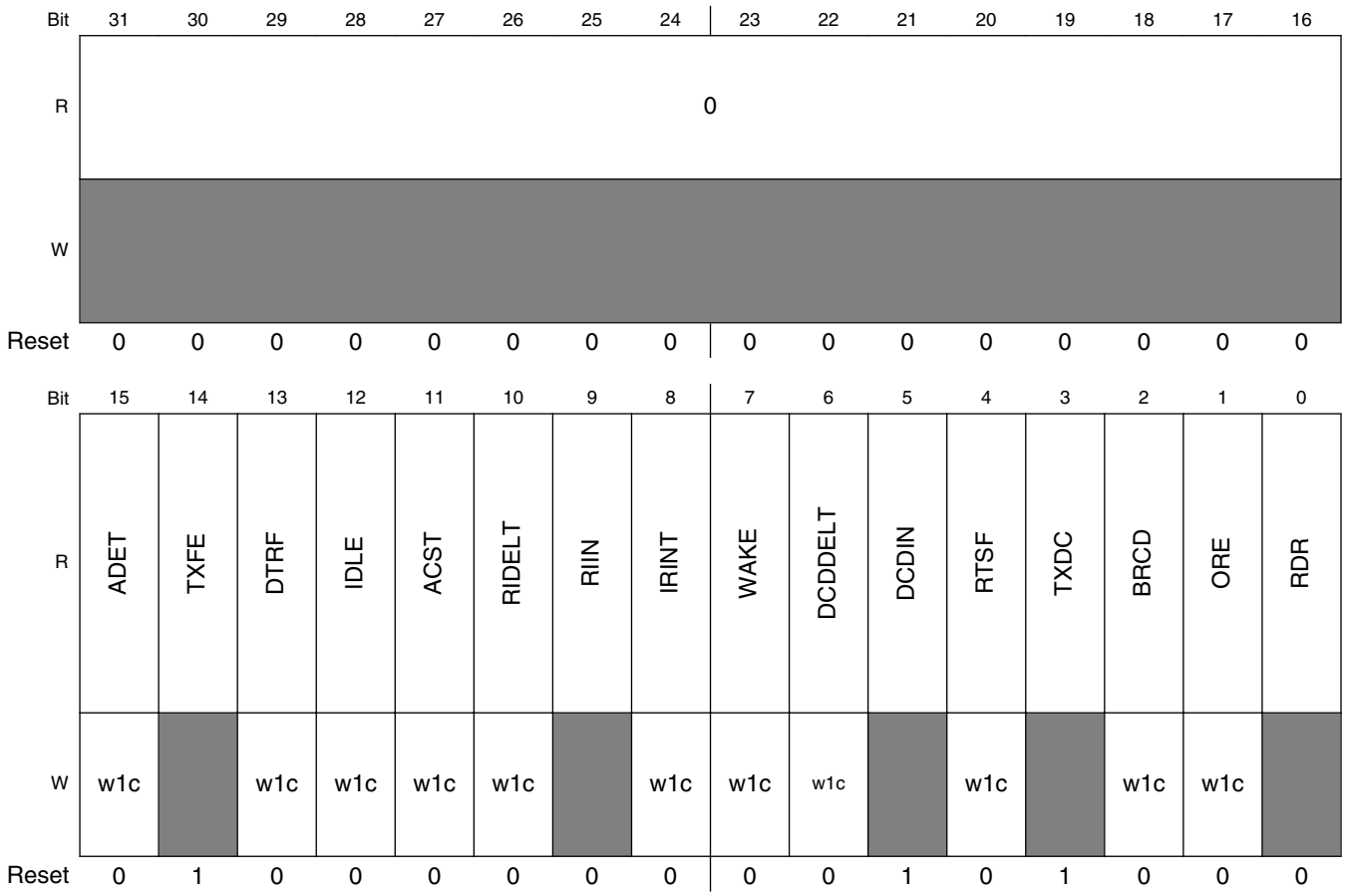
Table continues on the next page...

UARTx_USR1 field descriptions (continued)

Field	Description
	0 Receive in progress 1 Receiver is IDLE
5 AIRINT	Asynchronous IR WAKE Interrupt Flag. Indicates that the IR WAKE pulse was detected on the RXD pin. Clear AIRINT by writing 1 to it. Writing 0 to AIRINT has no effect. 0 No pulse was detected on the RXD IrDA pin 1 A pulse was detected on the RXD IrDA pin
4 AWAKE	Asynchronous WAKE Interrupt Flag. Indicates that a falling edge was detected on the RXD pin. Clear AWAKE by writing 1 to it. Writing 0 to AWAKE has no effect. 0 No falling edge was detected on the RXD Serial pin 1 A falling edge was detected on the RXD Serial pin
3 SAD	RS-485 Slave Address Detected Interrupt Flag. Indicates if RS-485 Slave Address was detected . SAD was asserted in RS-485 mode when the SADEN bit is set and Slave Address is detected in RxFIFO (in Nomal Address Detect Mode, the 9th data bit = 1; in Automatic Address Detect Mode, the received charater matches the programmed SLADDR). 0 No slave address detected 1 Slave address detected
-	This field is reserved. Reserved

64.15.9 UART Status Register 2 (UARTx_USR2)

Address: Base address + 98h offset



UARTx_USR2 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15 ADET	**Automatic Baud Rate Detect Complete**. Indicates that an "A" or "a" was received and that the receiver detected and verified the incoming baud rate. Clear ADET by writing 1 to it. Writing 0 to ADET has no effect. 0 ASCII "A" or "a" was not received 1 ASCII "A" or "a" was received (write 1 to clear)
14 TXFE	**Transmit Buffer FIFO Empty**. Indicates that the transmit buffer (TxFIFO) is empty. TXFE is cleared automatically when data is written to the TxFIFO. Even though TXFE is high, the transmission might still be in progress. 0 The transmit buffer (TxFIFO) is not empty 1 The transmit buffer (TxFIFO) is empty

Table continues on the next page...

UARTx_USR2 field descriptions (continued)

Field	Description
13 DTRF	**DTR edge triggered interrupt flag**. This bit is asserted, when the programmed edge is detected on the DTR_B pin (DCE mode) or on DSR_B (DTE mode). This flag can cause an interrupt if DTREN (UCR3[13]) is enabled. 0 Programmed edge not detected on DTR/DSR 1 Programmed edge detected on DTR/DSR (write 1 to clear)
12 IDLE	**Idle Condition**. Indicates that an idle condition has existed for more than a programmed amount frame (see Idle Line Detect. An interrupt can be generated by this IDLE bit if IDEN (UCR1[12]) is enabled. IDLE is cleared by writing 1 to it. Writing 0 to IDLE has no effect. 0 No idle condition detected 1 Idle condition detected (write 1 to clear)
11 ACST	**Autobaud Counter Stopped**. In autobaud detection (ADBR=1), indicates the counter which determines the baud rate was running and is now stopped. This means either START bit is finished (if ADNIMP=1), or Bit 0 is finished (if ADNIMP=0). See New Autobaud Counter Stopped bit and Interrupt, for more details. An interrupt can be flagged on *interrupt_uart* if ACIEN=1. 0 Measurement of bit length not finished (in autobaud) 1 Measurement of bit length finished (in autobaud). (write 1 to clear)
10 RIDELT	**Ring Indicator Delta**. This bit is used in DTE mode to indicate that the Ring Indicator input (RI_B) has changed state. This flag can generate an interrupt if RI (UCR3[8]) is enabled. RIDELT is cleared by writing 1 to it. Writing 0 to RIDELT has no effect. 0 Ring Indicator input has not changed state 1 Ring Indicator input has changed state (write 1 to clear)
9 RIIN	**Ring Indicator Input**. This bit is used in DTE mode to reflect the status if the Ring Indicator input (RI_B). The Ring Indicator input is used to indicate that a ring has occurred. In DCE mode this bit is always zero. 0 Ring Detected 1 No Ring Detected
8 IRINT	**Serial Infrared Interrupt Flag.** When an edge is detected on the RXD pin during SIR Mode, this flag will be asserted. This flag can cause an interrupt which can be masked using the control bit ENIRI: UCR4 [8]. 0 no edge detected 1 valid edge detected (write 1 to clear)
7 WAKE	**Wake**. Indicates the start bit is detected. WAKE can generate an interrupt that can be masked using the WKEN bit. Clear WAKE by writing 1 to it. Writing 0 to WAKE has no effect. 0 start bit not detected 1 start bit detected (write 1 to clear)
6 DCDDELT	**Data Carrier Detect Delta**. This bit is used in DTE mode to indicate that the Data Carrier Detect input (DCD_B) has changed state. This flag can cause an interrupt if DCD (UCR3[9]) is enabled. When in STOP mode, this bit can be used to wake the processor. In DCE mode this bit is always zero. 0 Data Carrier Detect input has not changed state 1 Data Carrier Detect input has changed state (write 1 to clear)
5 DCDIN	**Data Carrier Detect Input**. This bit is used in DTE mode reflect the status of the Data Carrier Detect input (DCD_B). The Data Carrier Detect input is used to indicate that a carrier signal has been detected. In DCE mode this bit is always zero.

Table continues on the next page...

UARTx_USR2 field descriptions (continued)

Field	Description
	0 Carrier signal Detected 1 No Carrier signal Detected
4 RTSF	**RTS Edge Triggered Interrupt Flag. Indicates if** a programmed edge is detected on the RTS_B pin. The RTEC bits select the edge that generates an interrupt (see Table 64-8). RTSF can generate an interrupt that can be masked using the RTSEN bit. Clear RTSF by writing 1 to it. Writing 0 to RTSF has no effect. 0 Programmed edge not detected on RTS_B 1 Programmed edge detected on RTS_B (write 1 to clear)
3 TXDC	**Transmitter Complete**. Indicates that the transmit buffer (TxFIFO) and Shift Register is empty; therefore the transmission is complete. TXDC is cleared automatically when data is written to the TxFIFO. 0 Transmit is incomplete 1 Transmit is complete
2 BRCD	**BREAK Condition Detected**. Indicates that a BREAK condition was detected by the receiver. Clear BRCD by writing 1 to it. Writing 0 to BRCD has no effect. 0 No BREAK condition was detected 1 A BREAK condition was detected (write 1 to clear)
1 ORE	**Overrun Error**. When set to 1, ORE indicates that the receive buffer (RxFIFO) was full (32 chars inside), and a 33rd character has been fully received. This 33rd character has been discarded. Clear ORE by writing 1 to it. Writing 0 to ORE has no effect. 0 No overrun error 1 Overrun error (write 1 to clear)
0 RDR	**Receive Data Ready**-Indicates that at least 1 character is received and written to the RxFIFO. If the URXD register is read and there is only 1 character in the RxFIFO, RDR is automatically cleared. 0 No receive data ready 1 Receive data ready

64.15.10 UART Escape Character Register (UARTx_UESC)

Address: Base address + 9Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																ESC_CHAR															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	1

UARTx_UESC field descriptions

Field	Description
31–8 Reserved	This read-only field is reserved and always has the value 0.
ESC_CHAR	**UART Escape Character**. Holds the selected escape character that all received characters are compared against to detect an escape sequence.

64.15.11 UART Escape Timer Register (UARTx_UTIM)

Address: Base address + A0h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																TIM															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

UARTx_UTIM field descriptions

Field	Description
31–12 Reserved	This read-only field is reserved and always has the value 0.
TIM	UART Escape Timer. Holds the maximum time interval (in ms) allowed between escape characters. The escape timer register is programmable in intervals of 2 ms. See Escape Sequence Detection and Table 64-13 for more information on the UART escape sequence detection. Reset value 0x000 = 2 ms up to 0xFFF = 8.192 s.

64.15.12 UART BRM Incremental Register (UARTx_UBIR)

This register can be written by both software and hardware. When enabling the automatic baud rate detection feature hardware can write 0x000F value into the UBIR after finishing detecting baud rate. Hardware has higher priority when both software and hardware try to write it at the same cycle[3].

Please note software reset will reset the register to its reset value.

Address: Base address + A4h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																INC															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

UARTx_UBIR field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
INC	Incremental Numerator. Holds the numerator value minus one of the BRM ratio (see Binary Rate Multiplier (BRM)). The UBIR register MUST be updated before the UBMR register for the baud rate to be updated correctly. If only one register is written to by software, the BRM will ignore this data until the other register is written to by software. Updating this field using byte accesses is not recommended and is undefined.

3. Note: The write priority in the new design is not same as the original UART. In the orginal design, software has higher priority than hardware when writing this register at the same time.

64.15.13 UART BRM Modulator Register (UARTx_UBMR)

This register can be written by both software and hardware. When enabling the automatic baud rate detection feature hardware can write a proper value into the UBMR based on detected baud rate. Hardware has higher priority when both software and hardware try to write it at the same cycle[4].

Please note software reset will reset the register to its reset value.

Address: Base address + A8h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																MOD															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

UARTx_UBMR field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
MOD	**Modulator Denominator.** Holds the value of the denominator minus one of the BRM ratio (see Binary Rate Multiplier (BRM)). The UBIR register MUST be updated before the UBMR register for the baud rate to be updated correctly. If only one register is written to by software, the BRM will ignore this data until the other register is written to by software. Updating this register using byte accesses is not recommended and undefined.

64.15.14 UART Baud Rate Count Register (UARTx_UBRC)

Address: Base address + ACh offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																BCNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

UARTx_UBRC field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
BCNT	**Baud Rate Count Register.** This read only register is used to count the start bit of the incoming baud rate (if ADNIMP=1), or start bit + bit0 (if ADNIMP=0). When the measurement is done, the Baud Rate Count Register contains the number of UART internal clock cycles (clock after divider) present in an incoming bit. BCNT retains its value until the next Automatic Baud Rate Detection sequence has been initiated. The 16 bit Baud Rate Count register is reset to 4 and stays at hex FFFF in the case of an overflow.

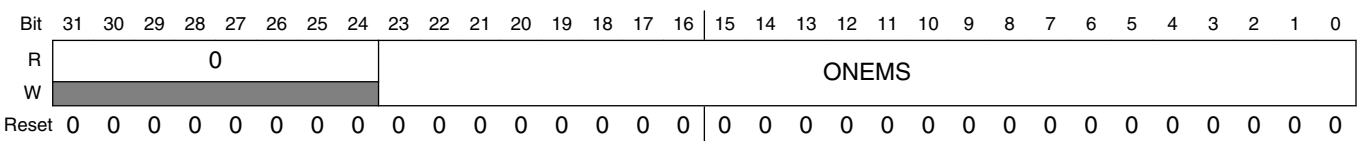
4. Note: The write priority in the new design is not same as the original UART. In the orginal design, software has higher priority than hardware when writing this register at the same time.

64.15.15 UART One Millisecond Register (UARTx_ONEMS)

NOTE

This register has been expanded from 16 bits to 24 bits. In previous versions, the 16-bit ONEMS can only support the maximum 65.535MHz (0xFFFFx1000) *ref_clk*. To support 4Mbps Bluetooth application with 66.5MHz *module_clock*, the value 0x103C4 (66.5M/1000) should be written into this register. In this case, the 16 bits are not enough to contain the 0x103C4. So this register was expanded to 24 bits to support high frequency of the *ref_clk*.

Address: Base address + B0h offset

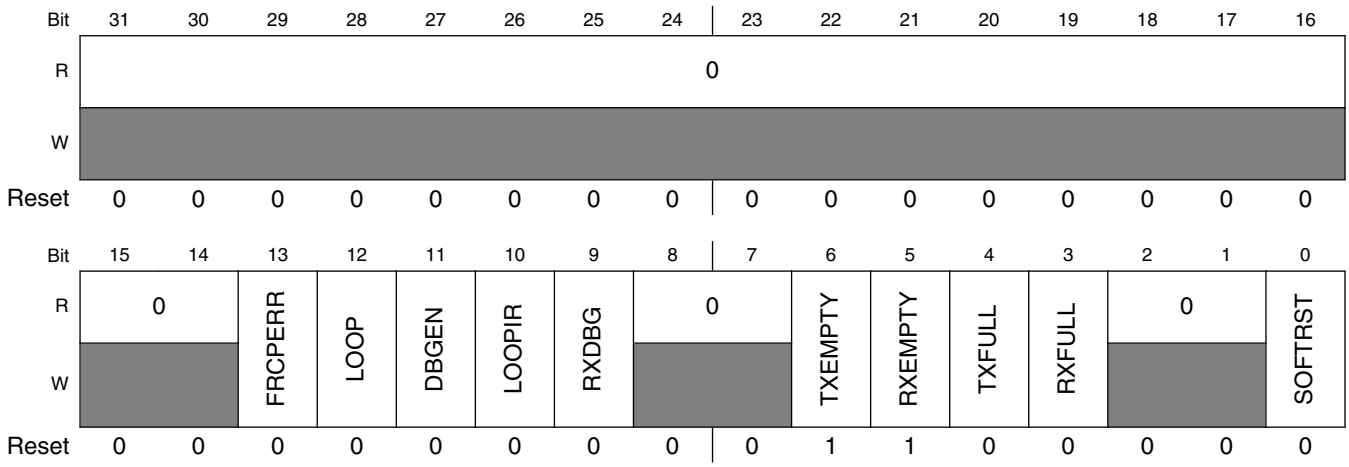


UARTx_ONEMS field descriptions

Field	Description
31–24 Reserved	This read-only field is reserved and always has the value 0.
ONEMS	**One Millisecond Register.** This 24-bit register must contain the value of the UART internal frequency (*ref_clk* in Figure 64-1) divided by 1000. The internal frequency is obtained after the UART BRM internal divider (F (*ref_clk*) = F(*module_clock*) / RFDIV). In fact this register contains the value corresponding to the number of UART BRM internal clock cycles present in one millisecond. The ONEMS (and UTIM) registers value are used in the escape character detection feature (Escape Sequence Detection) to count the number of clock cycles left between two escape characters. The ONEMS register is also used in infrared special case mode (IRSC = UCR4[5] = 1'b1), see InfraRed Special Case (IRSC) Bit.

64.15.16 UART Test Register (UARTx_UTS)

Address: Base address + B4h offset



UARTx_UTS field descriptions

Field	Description
31–14 Reserved	This read-only field is reserved and always has the value 0.
13 FRCPERR	Force Parity Error. Forces the transmitter to generate a parity error if parity is enabled. FRCPERR is provided for system debugging. 0 Generate normal parity 1 Generate inverted parity (error)
12 LOOP	Loop TX and RX for Test. Controls loopback for test purposes. When LOOP is high, the receiver input is internally connected to the transmitter and ignores the RXD pin. The transmitter is unaffected by LOOP. If RXDMUXSEL (UCR3[2]) is set to 1, the loopback is applied on serial and IrDA signals. If RXDMUXSEL is set to 0, the loopback is only applied on serial signals. 0 Normal receiver operation 1 Internally connect the transmitter output to the receiver input
11 DBGEN	debug_enable_B**.** This bit controls whether to respond to the *debug_req* input signal. 0 UART will go into debug mode when *debug_req* is HIGH 1 UART will not go into debug mode even if *debug_req* is HIGH
10 LOOPIR	**Loop TX and RX for IR Test (LOOPIR)**. This bit controls loopback from transmitter to receiver in the InfraRed interface. 0 No IR loop 1 Connect IR transmitter to IR receiver
9 RXDBG	**RX_fifo_debug_mode.** This bit controls the operation of the RX fifo read counter when in debug mode. 0 rx fifo read pointer does not increment 1 rx_fifo read pointer increments as normal

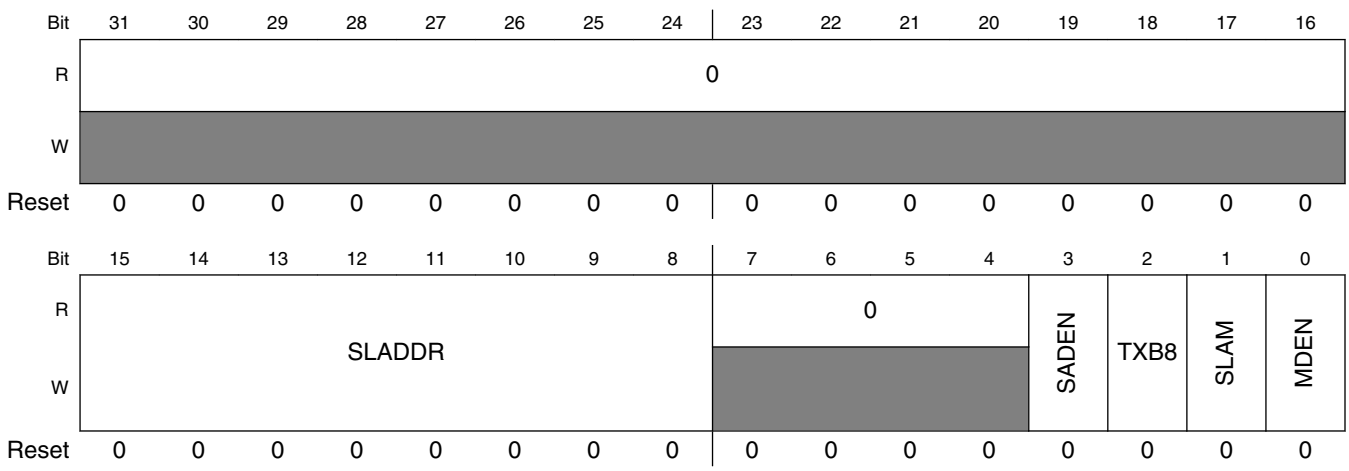
Table continues on the next page...

UARTx_UTS field descriptions (continued)

Field	Description
8–7 Reserved	This read-only field is reserved and always has the value 0.
6 TXEMPTY	TxFIFO Empty. Indicates that the TxFIFO is empty. 0 The TxFIFO is not empty 1 The TxFIFO is empty
5 RXEMPTY	RxFIFO Empty. Indicates the RxFIFO is empty. 0 The RxFIFO is not empty 1 The RxFIFO is empty
4 TXFULL	TxFIFO FULL. Indicates the TxFIFO is full. 0 The TxFIFO is not full 1 The TxFIFO is full
3 RXFULL	RxFIFO FULL. Indicates the RxFIFO is full. 0 The RxFIFO is not full 1 The RxFIFO is full
2–1 Reserved	This read-only field is reserved and always has the value 0.
0 SOFTRST	Software Reset. Indicates the status of the software reset (SRST_B bit of UCR2). 0 Software reset inactive 1 Software reset active

64.15.17 UART RS-485 Mode Control Register (UARTx_UMCR)

Address: Base address + B8h offset



UARTx_UMCR field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15–8 SLADDR	RS-485 Slave Address Character. Holds the selected slave adress character that the receiver wil try to detect.
7–4 Reserved	This read-only field is reserved and always has the value 0.
3 SADEN	RS-485 Slave Address Detected Interrupt Enable. 0 Disable RS-485 Slave Address Detected Interrupt 1 Enable RS-485 Slave Address Detected Interrupt
2 TXB8	Transmit RS-485 bit 8 (the ninth bit or 9th bit). In RS-485 mode, software writes TXB8 bit as the 9th data bit to be transmitted. 0 0 will be transmitted as the RS485 9th data bit 1 1 will be transmitted as the RS485 9th data bit
1 SLAM	RS-485 Slave Address Detect Mode Selection. 0 Select Normal Address Detect mode 1 Select Automatic Address Detect mode
0 MDEN	9-bit data or Multidrop Mode (RS-485) Enable. 0 Normal RS-232 or IrDA mode, see Table 64-1 for detail. 1 Enable RS-485 mode, see Table 64-1 for detail

Chapter 65
Universal Serial Bus Controller (USB)

65.1 Overview

The USB controller block provides high performance USB functionality that conforms to the *Universal Serial Bus Specification*, Rev. 2.0 (Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, Philips; 2000), and the *On-The-Go and Embedded Host Supplement to the USB Revision* 2.0 Specification (Hewlett-Packard Company, Intel Corporation, LSI Corporation, Microsoft Corporation, Renesas Electronics Corporation, ST-Ericsson; 2012).

The USB controller consists of four independent USB controller cores: one On-The-Go (OTG) controller core, and three host-only controller cores. Each controller core can support ULPI, Serial, UTMI, IC-USB or HSIC interface according to its feature.

See Features for more details. All four controller cores are single-port cores. For the OTG core, there is only one port. It can be used as either a downstream or an upstream port. For the host-only core, there is also only one port which is used as a downstream port.

The following figure is a block diagram of USB.

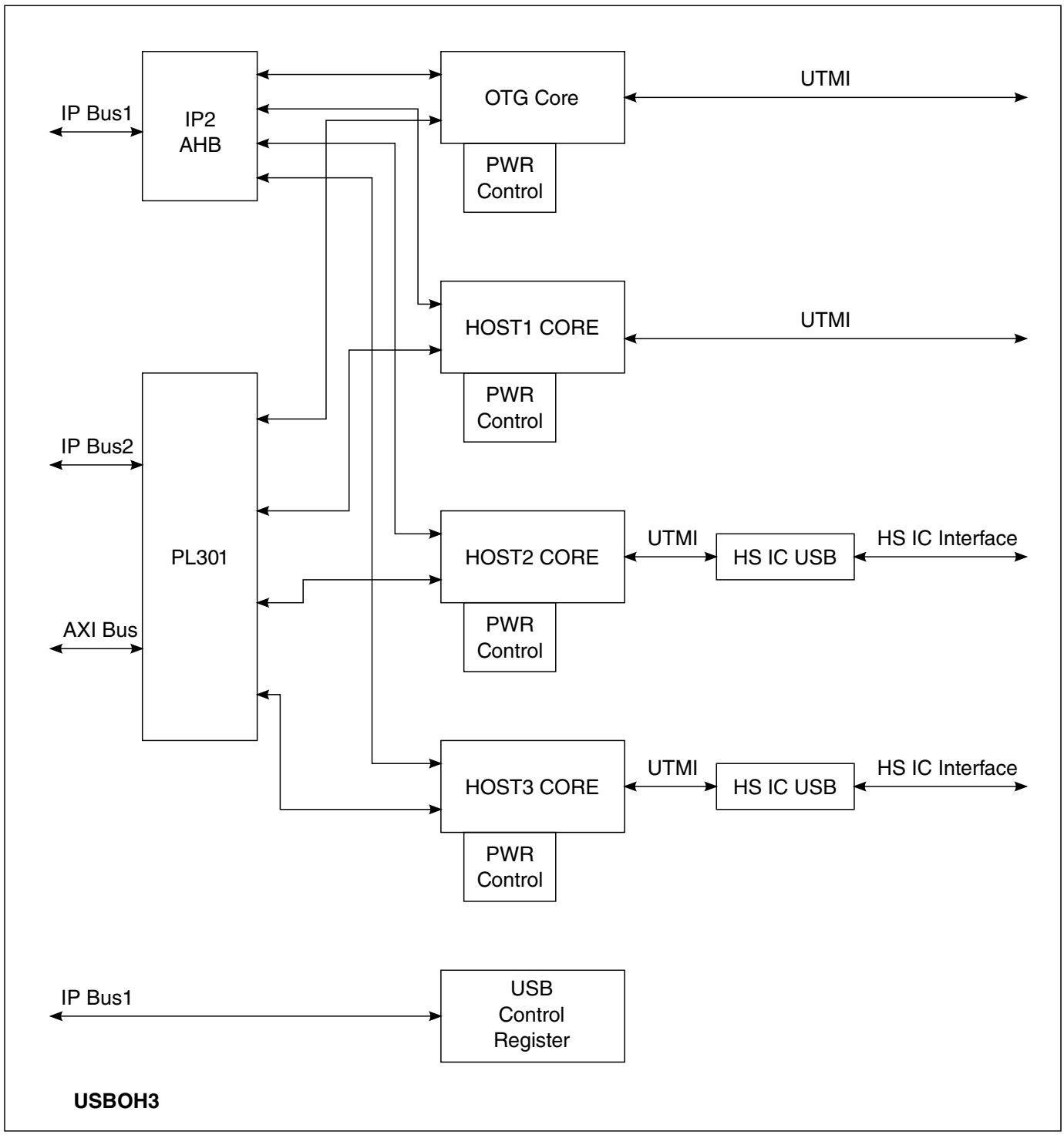


Figure 65-1. USB block diagram

65.1.1 Features

There are four USB 2.0 controller cores in this chip:

- Controller Core 0 is also named 'OTG Core'; its connected port is named 'OTG port'.
- Controller Core 1 is also named 'Host1 Core'; its connected port is named 'Host1 port'.
- Controller Core 2 is also named 'Host2 Core'; its connected port is named 'Host2 port'.
- Controller Core 3 is also named 'Host3 Core'; its connected port is named 'Host3 port'.

The following list provides features of each of the controller cores.

- USB 2.0 Controller Core 0
  - High-Speed/Full-Speed/Low-Speed OTG core
  - HS/FS/LS UTMI compliant interface
  - High Speed, Full Speed and Low Speed operation in Host mode (with UTMI transceiver)
  - High Speed, and Full Speed operation in Peripheral mode (with UTMI transceiver)
  - Hardware support for OTG signaling, session request protocol, and host negotiation protocol
  - Up to 8 bidirectional endpoints
  - Support charger detection
- USB 2.0 Controller Core 1
  - High-Speed/Full-Speed/Low-Speed Host-Only core
  - HS/FS/LS UTMI compliant interface
- USB 2.0 Controller Core 2
  - High-Speed/Full-Speed/Low-Speed Host-Only core
  - High Speed Inter-Chip USB compliant interface (HSIC)
- USB 2.0 Controller Core 3
  - High-Speed/Full-Speed/Low-Speed Host-Only core
  - High Speed Inter-Chip USB compliant interface (HSIC)
- Low-power mode with local and remote wake-up capability
- Serial PHY interfaces configurable for bidirectional/unidirectional and differential/single ended
- Embedded DMA controller in each core

65.1.2 Modes of Operation

The USB has two main modes of operation: normal mode and low power mode.

Each USB controller core can operate in High Speed operation (480 Mbps), Full Speed operation (12Mbps) and Low Speed operation (1.5 Mbps).

This chapter explains the operation modes.

65.1.2.1 Normal Mode

The OTG controller core can operate in Host mode and Device (Peripheral) mode. The host-only controller core can operate in Host mode only.

Each USB controller core has its corresponding port, which can work in one or more interface modes.

NOTE

Each controller supports only the interface type listed below. Selecting a different interface type in the PORTSC.PTS field results in unpredictable behavior and may cause the system to hang.

- OTG port
  - This port supports on-chip UTMI transceivers only.
- Host1 Port
  - This port supports on-chip UTMI transceivers only.
- Host2 Port
  - This port supports HSIC interfaces only.

    Interface for onboard HSIC compatible USB peripherals.
- Host3 Port
  - This port supports HSIC interface only.

    Interface for onboard HSIC compatible USB peripherals.

NOTE

HSIC is an inter-chip interface that is optimized for circuit board layouts.

65.1.2.2 Low-Power Mode

Each USB controller core has a low-power mode (Suspend mode) to save power consumption.

As described in the USB 2.0 specification, the device can go into the Suspend state after it sees a constant Idle state on the upstream facing port. The OTG controller core enters Suspend mode after 3 ms of inactivity on the port when it is in Device Operation mode. Host controllers, including the OTG controller in Host mode, do not suspend automatically but can be placed in Suspend mode by software.

Either the local ARM platform or the remote USB Host/Peripheral can initiate a wake-up sequence to resume USB communication. For details about Suspend/Resume, see USB Power Control.

65.2 External Signals

The table found here describes the external signals of USB.

Table 65-1. USB External Signals

Signal	Description	Pad	Mode	Direction
USB_H1_DN	DN Host 1 Signal	USB_H1_DN	No muxing	IO
USB_H1_DP	DP Host 1 Signal	USB_H1_DP	No muxing	IO
USB_H1_OC	Host 1 External input for VBUS overcurrent detection	EIM_D30	ALT6	I
		GPIO_3	ALT6	
USB_H1_PWR	To control PMIC to supply VBUS voltage	EIM_D31	ALT6	O
		GPIO_0	ALT6	
USB_H2_DATA	Data signal	RGMII_TXC	ALT0	IO
USB_H2_STROBE	Strobe signal	RGMII_TX_CTL	ALT0	IO
USB_H3_DATA	Data signal	RGMII_RX_CTL	ALT0	IO
USB_H3_STROBE	Strobe signal	RGMII_RXC	ALT0	IO
USB_OTG_CHD_B	Charge detect signal	USB_OTG_CHD_B	No muxing	IO
USB_OTG_DN	DN OTG Signal	USB_OTG_DN	No muxing	IO
USB_OTG_DP	DP OTG Signal	USB_OTG_DP	No muxing	IO
USB_OTG_ID	ID signal	ENET_RX_ER	ALT0	I
		GPIO_1	ALT3	
USB_OTG_OC	OTG External input for VBUS overcurrent detection	EIM_D21	ALT4	I
		KEY_COL4	ALT2	
USB_OTG_PWR	To control PMIC to supply VBUS voltage	EIM_D22	ALT4	O
		KEY_ROW4	ALT2	

65.3 Functional Description

These sections describe the functionality of the various building blocks of the USB.

65.3.1 USB 2.0 Controller Core 0

The USB 2.0 Controller 0 is an instantiation of an EHCI-compatible core which supports high-, full-, and low-speed operation.

In Host mode, this controller core supports high-, full-, and low-speed operation. In Device mode, it supports high- and full-speed operation.

65.3.1.1 Host Mode

The controller supports direct connection of a HS/FS/LS device with on-chip UTMI transceiver.

Although there is no separate Transaction Translator block in the system, the transaction translator function normally associated with a USB 2.0 high speed hub has been implemented within the DMA and protocol engine blocks to support connection to full and low speed devices.

65.3.1.2 Peripheral (Device) Mode

- Up to eight bidirectional endpoints
- High/full-speed operation
- Support of HNP and SRP
- Remote wake-up capability

65.3.1.3 Pins Used for OTG Controller

The required power supplies:

- 1.1-V supply from LDO 1P1 regulator
- 2.5-V supply from LDO 2P5 regulator
- 3.0-V supply from USB LDO regulator

USB OTG PHY pins:

- USB_OTG_VBUS
- USB_OTG_DN
- USB_OTG_DP
- USB_OTG_CHD_B, see Universal Serial Bus 2.0 Integrated PHY (USB-PHY).

The following external signals are multiplexed with other pins. For the pin mapping, see External Signals. For the IOMUXC register setting, see IOMUX Controller (IOMUXC)

- USB_OTG_ID
- USB_OTG_PWR
- USB_OTG_OC

65.3.2 USB 2.0 Controller Core 1

USB 2.0 Controller Core 1 is an instantiation of EHCI-compatible core which supports High Speed / Full Speed / Low Speed operation.

65.3.2.1 Pins Used for Host Controller 1

The required power supplies:

- 1.1V voltage supply from LDO 1P1 regulator
- 2.5V voltage supply from LDO 2P5 regulator
- 3.0V voltage supply from USB LDO regulator

USB Host 1 PHY pins:

- USB_H1_VBUS
- USB_H1_DN
- USB_H1_DP

The following external signals are multiplexed with other pins. For the pin mapping, see External Signals. For the IOMUXC register setting, see IOMUX Controller (IOMUXC)

- USB_H1_PWR
- USB_H1_OC

65.3.3 USB 2.0 Controller Core 2

USB 2.0 Controller Core 2 is an instantiation of the EHCI-compatible core which supports High Speed / Full Speed / Low Speed operation.

This USB core's signals connect directly to I/O pins (HSIC interface).

65.3.4 USB 2.0 Controller Core 3

Host Controller 3 is an instantiation of EHCI-compatible core which supports High Speed / Full Speed / Low Speed operation.

This USB core's signals connect directly to I/O pins (HSIC interface).

65.3.5 USB Power Control

The USB controller supports suspend and wake-up functionality.

The power control block allows for placing the transceiver in USB low power mode when USB bus is IDLE, and supports local and remote wake-up to bring the transceiver out of USB low power mode when needed. Additionally, the power control block can wake-up the ARM platform from core sleep mode by generating an interrupt.

65.3.5.1 Entering Low Power Suspend Mode

In Host operation mode, low power suspend mode is entered as follows:

1. Clear the ASE and PSE bits in USB_USBCMD, and wait until the AS and PS bits in USB_USBSTS become "0".
2. Set the "SUSPEND" bit in USB_PORTSC1.
3. Set the "PHCD" bit in USB_PORTSC1.
4. Set all PWD bits in USBPHYx_PWD
5. Set CLKGATE in USBPHYx_CTRL

NOTE

Step 3,4,5 shall be done in atomic operation. That is, interrupt should be disabled during these three steps.

For device operation mode, low power suspend mode is entered as follows:

1. After Host drive is IDLE for 3ms, an SLI interrupt is issued (the "DCSUSPEND" or "SLI" bit in USB_USBSTS).
2. Set the "PHCD" bit on USB_PORTSC1
3. Set all PWD bits in PHYPWD
4. Set CLKGATE in PHYCTRL

NOTE

Step 2,3,4 shall be done in atomic operation. That is, interrupt should be disabled during these three steps.

65.3.5.2 Wake-Up Events

The power control block monitors the USB bus when the USB core is in the USB suspend state.

Depending on whether the core is on Host or Device mode, a number of wake-up conditions are monitored. Upon detection of a wake-up condition, an interrupt (asynchronous) will be generated to ARM platform if the related wake-up interrupt enable bit is set.

USB wake-up interrupt also re-activates the ARM platform clocks if they were stopped during the suspend.

65.3.5.2.1 Host Mode Events

The host controller wakes up on the following events:

- Remote Wake-up Request

A peripheral can request the host to reactivate the bus by driving wake-up signaling on the DM/DP lines. The power control block sends a wake-up request to the USB core when a J-K transition on DM/DP line is detected.

- Wake-Up On Overcurrent

If Wake-Up On Overcurrent is enabled (WKOC bit in the USB core register PORTSC1 is set '1'), the power control block sends a wake-up request to the USB core when an overcurrent event is detected.

- Wake-Up On Disconnect

If Wake-Up On Disconnect is enabled (WKDC bit in the USB core register PORTSC1 is set '1'), the power control block sends a wake-up request to the USB core when a disconnection event is detected (J-SE0/K-SE0 transition on DM/DP line).

- Wake-Up On Connect

If a Wake-Up On Connect is enabled (WKCN bit in the USB core register PORTSC1 is set '1'), the power control sub-block sends a wake-up request to the USB core when the connection event is detected (SE0-J/SE0-K transition on DM/DP line).

For a detailed description of register bits WKOC, WKDC, WKCN, please see Port Status & Control (USBC_*n*_PORTSC1).

65.3.6 Interrupts

65.3.6.1 USB Core Interrupts

Each USB core uses one dedicated vector in the Interrupt Table. The vector numbers associated with each of the cores can be found in the Interrupt section.

With the exception of the wake-up interrupts, all of the interrupt sources are controlled in the USB Cores. Refer to the Interrupt Enable Register (USBC_*n*_USBINTR) for details.

65.3.6.2 USB Wake-Up Interrupts

Each USB Core has an associated wake-up interrupt. The wake-up interrupts are generated outside of the USB controller cores, but using the same vector as the corresponding USB controller cores interrupt.

These interrupts are generated by the Power Control blocks which run on the 32KHz standby clock. The wake-up interrupt is designed to work even when the USB and ARM platform clocks are disabled, such that a wake-up condition on the USB bus can re-activate the ARM platform clocks.

Because the wake-up interrupt is generated and cleared on a 32 KHz clock, this interrupt request responds very slowly to clear actions. For this reason, the software must disable the wake-up interrupt to clear the request flag. Disabling the interrupt masks the request instantaneously as this is clocked by the ARM platform clock. The software should than wait for at least three 32 KHz clock cycles before re-enabling this interrupt to allow sufficient time for the request flag to clear. Because this interrupt is only used during low power modes of the USB, it is sufficient to enable the wake-up interrupt just prior to entering the USB suspend mode.

65.4 USB Operation Model

This section describes the detailed application knowledge for Host1, Host2, Host3 and OTG ports.

It can be generally divided in two parts, one is for Host and the other is for Device. Host port applies to all host ports, and to OTG port when operating in Host mode. Device part only applies to OTG port when operating in Device mode.

65.4.1 Register Interface

Configuration, control and status registers are divided into three categories, identification, capability and operational registers.

NOTE

USB controller registers support only DWORD (32-bit) access.

- Identification registers are used to declare the slave interface presence along with the complete set of the hardware configuration parameters.
- Static, read only capability registers define the software limits, restrictions, and capabilities of the host/device controller.
- Operational registers are dynamic control or status registers that may be read only, read/write, or read/write-to-clear. The following sections define the use of these registers.

EHCI registers are listed alongside device registers to show the complementary nature of host and device control.

The following table describes the Interface register sets.

Table 65-2. Interface Register Sets

Offset	Register Set	Explanation
000h-07Ch	Identification Registers	Identification registers are used to declare the slave interface presence and include a table of the hardware configuration parameters.
100h-124h	Capability Registers	Capability registers specify the limits, restrictions, and capabilities of a host/device controller implementation. These values are used as parameters to the host/device controller driver.
080h-0FCh 140h-1FCh	Operational Registers	Operational registers are used by the system software to control and monitor the operational state of the host/device controller.

65.4.1.1 Configuration, Control and Status Register Set

The following table describes the Device/Host capability registers.

NOTE

Depending on implementation, "*x*" can have the following values: UOG, UH1, UH2 or UH3.

.

Table 65-3. Device/Host Capability Registers

Offset	Size (Bytes)	Mnemonic	Register Name	Device Mode	Host Mode
000h	4	USB_*x*_ID	Identification Register	O	O
004h	4	USB_*x*_HWGENERAL	General Hardware Parameters	O	O
008h	4	USB_*x*_HWHOST	Host Hardware Parameters	X	O
00Ch	4	USB_*x*_HWDEVICE	Device Hardware Parameters	O	X
010h	4	USB_*x*_HWTXBUF	TX Buffer Hardware Parameters	O	O
014h	4	USB_*x*_HWRXBUF	RX Buffer Hardware Parameters	O	O
018-07Fh		-	Reserved		
080h	4	USB_*x*_GPTIMER0LD	General Purpose Timer #0 Load Register	O	O
084h	4	USB_*x*_GPTIMER0CTRL	General Purpose Timer #0 Control Register	O	O
088h	4	USB_*x*_GPTIMER1LD	General Purpose Timer #1 Load Register	O	O
08Ch	4	USB_*x*_GPTIMER1CTRL	General Purpose Timer #1 Control Register	O	O
090h	4	USB_*x*_SBUSCFG	System Bus Interface Configuration Register	O	O
094-09Fh		-	Reserved		
100h	1	USB_*x*_CAPLENGTH	Capability Register Length	O	O
101h		-	Reserved		
102h	2	USB_*x*_HCIVERSION	Host Controller Interface Version Number	X	O
104h	4	USB_*x*_HCSPARAMS	Host Controller Structural Parameters	X	O
108h	4	USB_*x*_HCCPARAMS	Host Controller Capability Parameters	X	O
10C-11Fh		-	Reserved		
120h	2	USB_*x*_DCIVERSION	Device Controller Interface Version Number	O	X
122h	2	-	Reserved		
124h	4	USB_*x*_DCCPARAMS	Device Controller Capability Parameters	O	X
128-13Fh		-	Reserved		
140h	4	USB_*x*_USBCMD	USB Command Register	O	O
144h	4	USB_*x*_USBSTS	USB Status Register	O	O
148h	4	USB_*x*_USBINTR	USB Interrupt Enable Register	O	O
14Ch	4	USB_*x*_FRINDEX	USB Frame Index	O	O
150h	4	-	Reserved		
154h	4	USB_*x*_PERIODICLISTBASE	Frame List Base Address	X	O
		USB_*x*_DEVICEADDR	USB Device Address	O	X
158h	4	USB_*x*_ASYNCLISTADDR	Next Asynchronous List Address	X	O
	4	USB_*x*_ENDPOINTLISTADDR	Address at Endpoint list in memory	O	X
15Ch	4	-	Reserved		
160h	4	USB_*x*_BURSTSIZE	Programmable Burst Size	O	O
164h	4	USB_*x*_TXFILLTUNING	Host Transmit Pre-Buffer Packet Tuning	X	O

Table continues on the next page...

Table 65-3. Device/Host Capability Registers (continued)

Offset	Size (Bytes)	Mnemonic	Register Name	Device Mode	Host Mode
168h	4	-	Reserved		
16Ch	4	USB_*x*_IC_USB	IC_USB enable and voltage negotiation	O	O
170h	4	-	Reserved		
174h	4	USB_*x*_ENDPTNAK	Endpoint NAK register	O	X
178h	4	USB_*x*_ENDPTNAKEN	Endpoint NAK Enable register	O	X
17Ch	4	-	Reserved		
180h	4	USB_*x*_CONFIGFLAG	Configured Flag Register	X	O
184h	4	USB_*x*_PORTSC1	Port Status/Control Register 1	O	O
188-1A3h		-	Reserved		
1A4h	4	USB_*x*_OTGSC	On-The-Go Status/Control Register (OTG only)	O	O
1A8h	4	USB_*x*_USBMODE	USB Controller Operating Mode	O	O
1ACh	4	USB_*x*_ENDPTSETUPSTAT	Endpoint Setup Status	O	X
1B0h	4	USB_*x*_ENDPTPRIME	Endpoint Initialization	O	X
1B4h	4	USB_*x*_ENDPTFLUSH	Endpoint De-Initialization	O	X
1B8h	4	USB_*x*_ENDPTSTATUS	Endpoint Status	O	X
1BCh	4	USB_*x*_ENDPTCOMPLETE	Endpoint Complete	O	X
1C0 1C4 ... 1DCh	64	USB_*x*_ENDPTCTRL0 USB_*x*_ENDPTCTRL1 .... USB_*x*_ENDPTCTRL7	Endpoint Control Register 0-7	O	X

NOTE

"O" means the register is available in host/device operation mode;

"X" means the register is reserved in host/device operation mode

65.4.1.2 Identification Registers

Identification registers are used to declare the slave interface presence and include a table of the hardware configuration parameters.

65.4.1.3 OTG Operations

65.4.1.3.1 Register Bits

In the previous section, the Register interface has behaviors described for device mode and behaviors described for host mode. However, for OTG operations it is necessary to perform tasks independent of the controller mode.

Note that the only way to transit the controller mode out of host or device mode is with the controller reset bit. Therefore, it is also necessary for the OTG tasks to be performed independent of a controller reset as well as independent of the controller mode.

The following figure shows the controller mode.

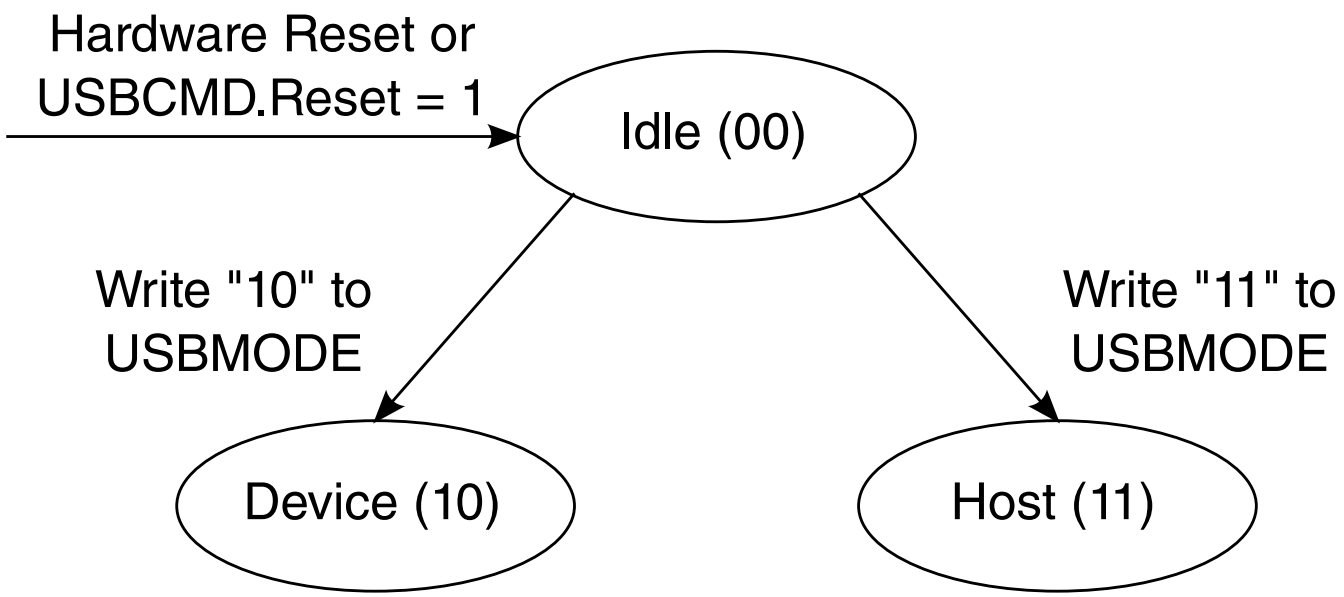


Figure 65-2. Controller Mode

To this end, listed below are the register bits that are used for OTG operations, which are independent of the controller mode and are also not affected by a write to the reset bit in the USBCMD register:

All Identification Registers

All Device/Host Capability Registers

OTGSC: All bits

PORTSC1:

Physical Interface Select

Physical Interface Serial Select

Physical Interface Data Width

Physical Interface Low Power

Physical Interface Wake Signals

Port Indicators

Port Power

65.4.2 Host Data Structures

This section defines the interface data structures used to communicate control, status, and data between HCD (software) and the Enhanced Host Controller (hardware).

The data structure definitions in this chapter support a 32-bit memory buffer address space. The interface consists of a Periodic Schedule, Periodic Frame List, Asynchronous Schedule, Isochronous Transaction Descriptors, Split-transaction Isochronous Transfer Descriptors, Queue Heads, and Queue Element Transfer Descriptors.

The periodic frame list is the root of all periodic (isochronous and interrupt transfer type) transfers for the host controller. The asynchronous list is the root for all the bulk and control transfers. Isochronous data streams are managed using Isochronous Transaction Descriptors. Isochronous split-transaction data streams are managed with Split-transaction Isochronous Transfer Descriptors. All Interrupt, Control, and Bulk data streams are managed via queue heads and Queue Element Transfer Descriptors. These data structures are optimized to reduce the total memory footprint of the schedule and to reduce (on average) the number of memory accesses needed to execute a USB transaction.

Note that software must ensure that no interface data structure reachable by the EHCI host controller spans a 4 K-page boundary.

The data structures defined in this section are (from the host controller's perspective) a mix of read-only and read/writeable fields. The host controller must preserve the read-only fields on all data structure writes.

65.4.2.1 Periodic Frame List

This schedule is for all periodic transfers (isochronous and interrupt). The periodic schedule is referenced from the operational registers space using the USB_PERIODICLISTBASE address register and the USB_FRINDEX register.

The periodic schedule is based on an array of pointers called the Periodic Frame List.

The USB_PERIODICLISTBASE address register is combined with the USB_FRINDEX register to produce a memory pointer into the frame list. The Periodic Frame List implements a sliding window of work over time.

The following figure shows the organization of periodic schedule.

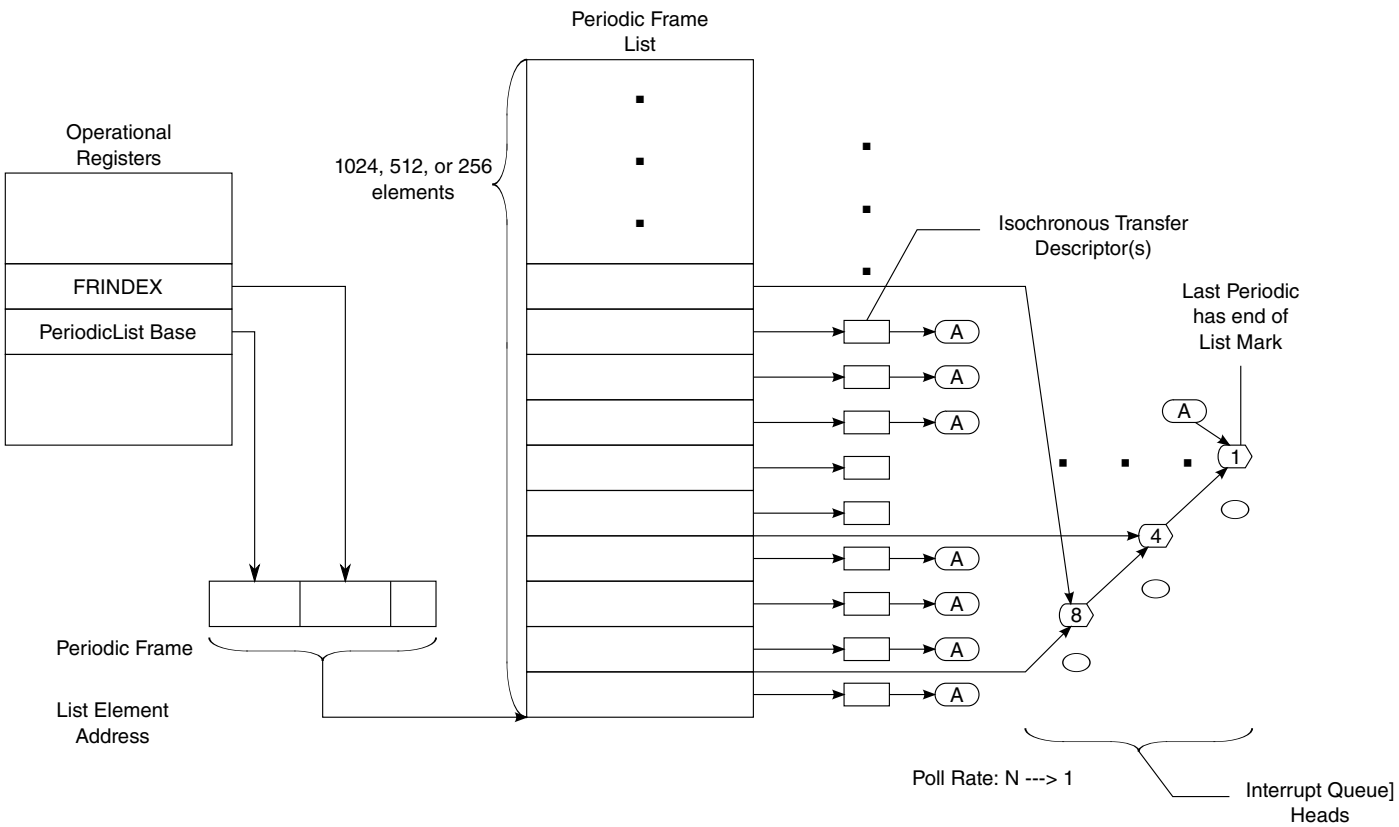


Figure 65-3. Periodic Schedule Organization

Split transaction Interrupt, Bulk and Control are also managed using queue heads and queue element transfer descriptors.

The periodic frame list is a 4 K-page aligned array of Frame List Link pointers. The length of the frame list may be programmable. The programmability of the periodic frame list is exported to system software via the USB_HCCPARAMS register. If non-programmable, the length is 1024 elements. If programmable, the length can be selected by system software as one of 256, 512, or 1024 elements. An implementation must support all three sizes. Programming the size (that is, the number of elements) is accomplished by system software writing the appropriate value into Frame List Size field in the USB_USBCMD register.

Frame List Link pointers direct the host controller to the first work item in the frame's periodic schedule for the current micro-frame. The link pointers are aligned on DWord boundaries within the Frame List.

The table below illustrates the format of the Frame list element pointer.

Table 65-4. Format of Frame List Element Pointer

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Frame List Link Pointer																											0		Typ			03-00H

Frame List Link pointers always reference memory objects that are 32-byte aligned. The referenced object may be an isochronous transfer descriptor for high-speed devices, a split-transaction isochronous transfer descriptor (for full-speed isochronous endpoints), or a queue head (used to support high-, full- and low-speed interrupt). System software should not place non-periodic schedule items into the periodic schedule. The least significant bits in a frame list pointer are used to key the host controller as to the type of object the pointer is referencing.

The least significant bit is the T-Bit (bit 0). When this bit is set to a one, the host controller never uses the value of the frame list pointer as a physical memory pointer. The Typ field is used to indicate the exact type of data structure being referenced by this pointer. The value encodings are.

Table 65-5. Typ Field Value Definitions

Value	Meaning
00b	Isochronous Transfer Descriptor
01b	Queue Head
10b	Split Transaction Isochronous Transfer Descriptor.
11b	Frame Span Traversal Node.

65.4.2.2 Asynchronous List Queue Head Pointer

The Asynchronous Transfer List (based at the USB_ASYNCLISTADDR register) is where all of the control and bulk transfers are managed.

Host controllers use this list only when it reaches the end of the periodic list, the periodic list is disabled, or the periodic list is empty.

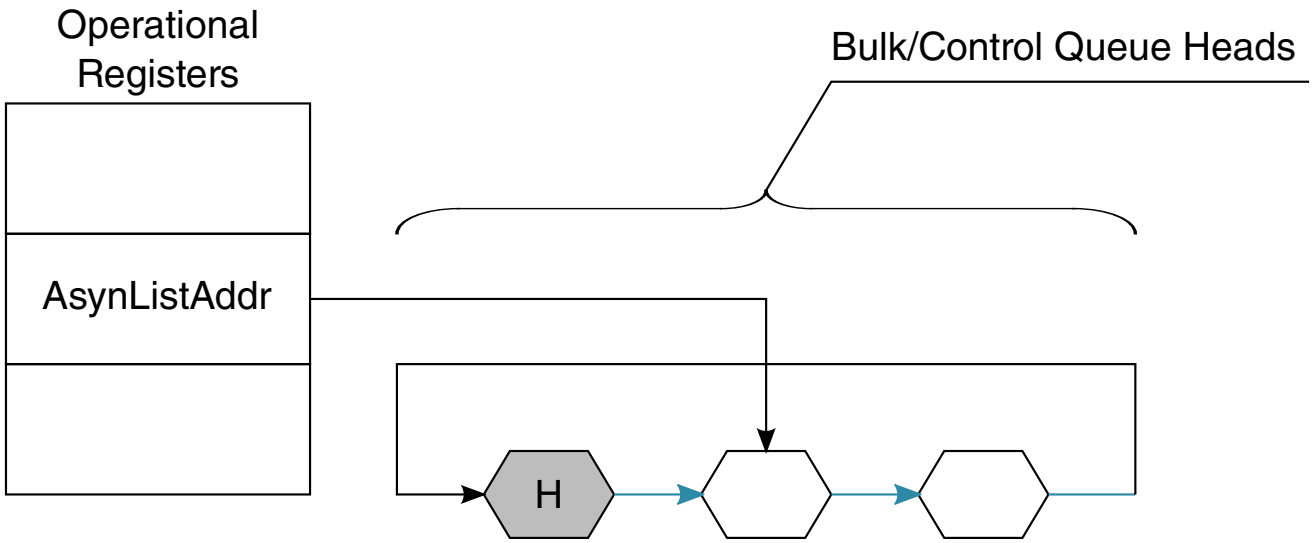


Figure 65-4. Asynchronous Schedule Organization

The Asynchronous list is a simple circular list of queue heads. The USB_ASYNCLISTADDR register is simply a pointer to the next queue head. This implements a pure round-robin service for all queue heads linked into the asynchronous list.

65.4.2.3 Isochronous (High-Speed) Transfer Descriptor (iTD)

The format of an isochronous transfer descriptor is shown in the table below.

This structure is used only for high-speed isochronous endpoints. All other transfer types should use queue structures. Isochronous TDs must be aligned on a 32-byte boundary.

Table 65-6. Isochronous Transaction Descriptor (iTD)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Next Link Pointer																											0		Typ		T	03-00 H
Status				Transaction 0 Length												IOC	PG*			Transaction 0 Offset*												07-04 H
Status				Transaction 1 Length												IOC	PG*			Transaction 1 Offset*												0B-08H
Status				Transaction 2 Length												IOC	PG*			Transaction 2 Offset*												0F-0CH
Status				Transaction 3 Length												IOC	PG*			Transaction 3 Offset*												13-10 H

Table continues on the next page...

Table 65-6. Isochronous Transaction Descriptor (iTD) (continued)

Status	Transaction 4 Length	IOC	PG*	Transaction 4 Offset*	17-14H
Status	Transaction 5 Length	IOC	PG*	Transaction 5 Offset*	1B-18H
Status	Transaction 6 Length	IOC	PG*	Transaction 6 Offset*	1F-1CH
Status	Transaction 7 Length	IOC	PG*	Transaction 7 Offset*	23-20H
Buffer Pointer (Page 0)				EndPt / R / Device Address	27-24H
Buffer Pointer (Page 1)				I/O / Maximum Packet Size	2B-28H
Buffer Pointer (Page 2)				- / Mult	2F-2CH
Buffer Pointer (Page 3)				-	33-30H
Buffer Pointer (Page 4)				-	37-34H
Buffer Pointer (Page 5)				-	3B-38H
Buffer Pointer (Page 6)				-	3F-3CH

Host Controller Read/Write

Host Controller Read Only

These fields may be modified by the host controller if the I/O field indicates an OUT.

65.4.2.3.1 Next Link Pointer

The first DWord of an iTD is a pointer to the next schedule data structure.

The following table describes the Next Schedule Element pointer field.

Table 65-7. Next Schedule Element Pointer

Bit	Description
31-5 Link Pointer (LP)	These bits correspond to memory address signals [31:5], respectively. This field points to another Isochronous Transaction Descriptor (iTD/siTD) or Queue Head (QH).
4-3	These bits are reserved and their value has no effect on operation. Software should initialize this field to zero.

Table continues on the next page...

Table 65-7. Next Schedule Element Pointer (continued)

Reserved	
2-1 QH/(s)iTD Select (Typ)	This field indicates to the Host Controller whether the item referenced is an iTD, siTD or a QH. This allows the Host Controller to perform the proper type of processing on the item after it is fetched. Value encodings are: Value Meaning 00b iTD (isochronous transfer descriptor) 01b QH (queue head) 10b siTD (split transaction isochronous transfer descriptor 11b FSTN (frame span traversal node)
0 Terminate (T)	1= Link Pointer field is not valid. 0= Link Pointer field is valid.

65.4.2.3.2 iTD Transaction Status and Control List

DWords 1 through 8 are eight slots of transaction control and status.

Each transaction description includes:

- Status results field
- Transaction length (bytes to send for OUT transactions and bytes received for IN transactions).
- Buffer offset. The PG and Transaction X Offset fields are used with the buffer pointer list to construct the starting buffer address for the transaction.

The host controller uses the information in each transaction description plus the endpoint information contained in the first three DWords of the Buffer Page Pointer list, to execute a transaction on the USB.

The following table describes iTD Transaction Status and Control fields.

Table 65-8. iTD Transaction Status and Control

Bit	Description	
31-28 Status	This field records the status of the transaction executed by the host controller for this slot. This field is a bit vector with the following encoding:	
	Bit	Definition
	31	Active. Set to one by software to enable the execution of an isochronous transaction by the Host Controller. When the transaction associated with this descriptor is completed, the Host Controller sets this bit to zero indicating that a transaction for this element should not be executed when it is next encountered in the schedule.
	30	Data Buffer Error. Set to a one by the Host Controller during status update to indicate that the Host Controller is unable to keep up with the reception of incoming data (overrun) or is unable to supply data fast enough during transmission (under run). If an overrun condition occurs, no action is necessary.
	29	Babble Detected. Set to one by the Host Controller during status update when" babble" is detected during the transaction generated by this descriptor.

Table continues on the next page...

Table 65-8. iTD Transaction Status and Control (continued)

Bit	Description	
	28	Transaction Error (XactErr). Set to one by the Host Controller during status update in the case where the host did not receive a valid response from the device (Timeout, CRC, Bad PID, etc.). This bit may only be set for isochronous IN transactions.
27-16 Transaction X Length	For an OUT, this field is the number of data bytes the host controller sends during the transaction. The host controller is not required to update this field to reflect the actual number of bytes transferred during the transfer. For an IN, the initial value of the endpoint to deliver. During the status update, the host controller writes back the field is the number of bytes the host expects the number of bytes successfully received. The value in this register is the actual byte count (0‡zero length data, 1‡one byte, 2‡two bytes, etc.). The maximum value this field may contain is 0xC00 (3072).	
15 Interrupt On Complete (IOC)	If this bit is set to one, it specifies that when this transaction completes, the Host Controller should issue an interrupt at the next interrupt threshold.	
14-12 Page Select (PG)	These bits are set by software to indicate which of the buffer page pointers the offset field in this slot should be concatenated to produce the starting memory address for this transaction. The valid range of values for this field is 0 to 6.	
11-0 Transaction X Offset	This field is a value that is an offset, expressed in bytes, from the beginning of a buffer. This field is concatenated onto the buffer page pointer indicated in the adjacent *PG* field to produce the starting buffer address for this transaction.	

65.4.2.3.3 iTD Buffer Page Pointer List (Plus)

DWords 9-15 of an isochronous transaction descriptor are nominally page pointers (4 K aligned) to the data buffer for this transfer descriptor. This data structure requires the associated data buffer to be contiguous (relative to virtual memory), but allows the physical memory pages to be non-contiguous.

Seven page pointers are provided to support the expression of eight isochronous transfers. The seven pointers allow for 3 (transactions) * 1024 (maximum packet size) * 8 (transaction records) (24576 bytes) to be moved with this data structure, regardless of the alignment offset of the first page.

Because each pointer is a 4 K aligned page pointer, the least significant 12 bits in several of the page pointers are used for other purposes.

The tables below illustrate the field descriptions.

Table 65-9. iTD Buffer Pointer Page 0 (Plus)

Bit	Description
31-12 Buffer Pointer (Page 0)	This is a 4 K aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11-8 Endpoint Number (Endpt)	This 4-bit field selects the particular endpoint number on the device serving as the data source or sink.

Table continues on the next page...

Table 65-9. iTD Buffer Pointer Page 0 (Plus) (continued)

Bit	Description
7 Reserved	Bit reserved for future use and should be initialized by software to zero.
6-0 Device Address	This field selects the specific device serving as the data source or sink.

Table 65-10. iTD Buffer Pointer Page 1 (Plus)

Bit	Description
31-12 Buffer Pointer (Page 1)	This is a 4K aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11 Direction (I/O)	0 = OUT; 1 = IN. This field encodes whether the high-speed transaction should use an IN or OUT PID.
10-0 Maximum Packet Size	This directly corresponds to the maximum packet size of the associated endpoint *(wMaxPacketSize)*. This field is used for high-bandwidth endpoints where more than one transaction is issued per transaction description (per micro-frame). This field is used with the *Multi* field to support high-bandwidth pipes. This field is also used for all IN transfers to detect packet babble. Software should not set a value larger than 1024 (400h). Any value larger yields undefined results.

Table 65-11. iTD Buffer Pointer Page 2 (Plus)

Bit	Description
31-12 Buffer Pointer	This is a 4K aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11-2 Reserved	This bit reserved for future use and should be set to zero.
1-0 Multi	This field is used to indicate to the host controller the number of transactions that should be executed per transaction description (per micro-frame). The valid values are: Value Meaning 00b Reserved. A zero in this field yields undefined results. 01b One transaction to be issued for this endpoint per micro- frame 10b Two transactions to be issued for this endpoint per micro- frame 11b Three transactions to be issued for this endpoint per micro- frame

Table 65-12. iTD Buffer Pointer Page 3-6

Bit	Description
31-12 Buffer Pointer	This is a 4 K aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11-0 Reserved	These bits reserved for future use and should be set to zero.

65.4.2.4 Split Transaction Isochronous Transfer Descriptor (siTD)

All Full-speed isochronous transfers through the internal transaction translator are managed using the siTD data structure. This data structure satisfies the operational requirements for managing the split transaction protocol.

The following table shows the Split Transaction Isochronous Transfer Descriptor (siTD).

Table 65-13. Split Transaction Isochronous Transfer Descriptor

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Addr
Next Link Pointer																											0		Typ		T	03-00
I/O	Port Number							-	Hub Addr							Reserved				EndPt				-	Device Address								07-04[1]
Reserved																µFrame C-mask								µFrame S-mask								0B-08[1]
ioc	P	Reserved				Total Bytes to Transfer										µFrame C-prog-mask								Status								0F-0C[2]
Buffer Pointer (Page 0)																				Current Offset												13-10[2]
Buffer Pointer (Page 1)																				Reserved							TP		T-count			17-14[2]
Back Pointer																											0				T	1B-18

1. 04-0B: Static Endpoint State
2. 0C-13: Transfer results

Host Controller Read/Write

Host Controller Read Only

65.4.2.4.1 Next Link Pointer

DWord0 of a siTD is a pointer to the next schedule data structure.

The following table describes the Next Link Pointer fields.

Table 65-14. Next Link Pointer

Bit	Description
31-5	Next Link Pointer (LP). This field contains the address of the next data object to be processed in the periodic list and corresponds to memory address signals [31:5], respectively.
4-3	Reserved. These bits must be written as zeros.

Table continues on the next page...

Table 65-14. Next Link Pointer (continued)

Bit	Description
2-1	QH/(s)iTD Select (Typ). This field indicates to the Host Controller whether the item referenced is an iTD/siTD or a QH. This allows the Host Controller to perform the proper type of processing on the item after it is fetched. Value encodings are: Value Meaning 00b iTD (isochronous transfer descriptor) 01b QH (queue head) 10b siTD (split transaction isochronous transfer descriptor 11b FSTN (frame span traversal node)
0	Terminate (T). 1=Link Pointer field is not valid. 0=Link Pointer is valid.

65.4.2.4.2 siTD Endpoint Capabilities/Characteristics

DWords 1 and 2 specify static information about the full-speed endpoint, the addressing of the parent Companion Controller, and micro-frame scheduling control.

The tables below describe the Endpoint and transaction translator characteristics and micro-frame schedule control fields.

Table 65-15. Endpoint and Transaction Translator Characteristics

Bit	Description
31	Direction (I/O).0 = OUT; 1 = IN. This field encodes whether the full-speed transaction should be an IN or OUT.
30-24	Port Number. This field is the port number of the recipient Transaction Translator.
23	Reserved. Bit reserved and should be set to zero.
22-16	Hub Address. This field holds the device address of the Companion Controllers' hub.
15-12	Reserved. Field reserved and should be set to zero.
11-8	Endpoint Number (Endpt). This 4-bit field selects the particular endpoint number on the device serving as the data source or sink.
7	Reserved. Bit is reserved for future use. It should be set to zero.
6-0	Device Address. This field selects the specific device serving as the data source or sink.

Table 65-16. Micro-frame Schedule Control

Bit	Description
31-16	Reserved. This field reserved for future use. It should be set to zero.
15-8	Split Completion Mask (mFrame C-Mask). This field (along with the *Active* and *SplitX- state* fields in the *Status* byte) is used to determine during which micro-frames the host controller should execute complete-split transactions. When the criteria for using this field is met, an all zeros value has undefined behavior. The host controller uses the value of the three low-order bits of the FRINDEX register to index into this bit field. If the FRINDEX register value indexes to a position where the m*Frame C-Mask* field is a one, then this siTD is a candidate for transaction execution. There may be more than one bit in this mask set.
7-0	Split Start Mask (mFrame S-mask). This field (along with the *Active* and *SplitX-state* fields in the *Status* byte) is used to determine during which micro-frames the host controller should execute start-split transactions. The host controller uses the value of the three low-order bits of the FRINDEX register to index into this bit field. If the FRINDEX register value indexes to a position where the m*Frame S-mask* field is a one, then this siTD is a candidate for transaction execution. An all zeros value in this field, in combination with existing periodic frame list has undefined results.

65.4.2.4.3 siTD Transfer State

DWords 3-6 are used to manage the state of the transfer.

The following table describes siTD transfer state fields.

Table 65-17. siTD Transfer Status and Control

Bit	Description
31	Interrupt On Complete (ioc). 0 = Do not interrupt when transaction is complete. 1 = Do interrupt when transaction is complete. When the host controller determines that the split transaction has completed it asserts a hardware interrupt at the next interrupt threshold.
30	Page Select (P). Used to indicate which data page pointer should be concatenated with the *CurrentOffset* field to construct a data buffer pointer (0 selects *Page 0* pointer and 1 selects *Page 1*). The host controller is not required to write this field back when the siTD is retired (*Active* bit transitioned from a one to a zero).
29-26	Reserved. This field reserved for future use and should be set to zero.
25-16	Total Bytes To Transfer. This field is initialized by software to the total number of bytes expected in this transfer. Maximum value is 1023 (3FFh)
15-8	µFrame Complete-split Progress Mask (C-prog-Mask). This field is used by the host controller to record which split-completes has been executed.
7-0: Status—This field records the status of the transaction executed by the host controller for this slot. It is a bit vector with the encoding shown in the following rows.	
7	Active. Set to one by software to enable the execution of an isochronous split transaction by the Host Controller.
6	ERR. Set to a one by the Host Controller when an ERR response is received from the Companion Controller.
5	Data Buffer Error. Set to a one by the Host Controller during status update to indicate that the Host Controller is unable to keep up with the reception of incoming data (overrun) or is unable to supply data fast enough during transmission (under run). In the case of an under run, the Host Controller transmits an incorrect CRC (thus invalidating the data at the endpoint). If an overrun condition occurs, no action is necessary.
4	Babble Detected. Set to a one by the Host Controller during status update when" babble" is detected during the transaction generated by this descriptor.
3	Transaction Error (XactErr). Set to a one by the Host Controller during status update in the case where the host did not receive a valid response from the device (Timeout, CRC, Bad PID, etc.). This bit is set only for IN transactions.
2	Missed Micro-Frame. The host controller detected that a host-induced hold- off caused the host controller to miss a required complete-split transaction.
1	Split Transaction State (SplitXstate). The bit encodings are: Value Meaning 00b Do Start Split. This value directs the host controller to issue a Start split transaction to the endpoint when a match is encountered in the S-mask. 01b Do Complete Split. This value directs the host controller to issue a Complete split transaction to the endpoint when a match is encountered in the C-mask.
0	Reserved. Bit reserved for future use and should be set to zero.

65.4.2.4.4 siTD Buffer Pointer List (plus)

DWords 4 and 5 are the data buffer page pointers for the transfer. This structure supports one physical page cross. The most significant 20 bits of each DWord in this section are the 4 K (page) aligned buffer pointers.

The least significant 12 bits of each DWord are used as additional transfer state. The following table describes the siTD buffer pointer fields.

Table 65-18. Buffer Page Pointer List (plus)

Bit	Description
31-12	Buffer Pointer List. Bits [31:12] of DWords 4 and 5 are 4 K page aligned physical memory addresses. These bits correspond to physical address bits [31:12] respectively. The lower 12 bits in each pointer are defined and used as specified below. The field *P* (see siTD Transfer State) specifies the *current* active pointer.
Bits 11-0 (Page 0)	Current Offset—The 12 least significant bits of the Page 0 pointer are the current byte offset for the current page pointer (as selected with the page indicator bit (*P* field). The host controller is not required to write this field back when the siTD is retired (*Active* bit transitioned from a one to a zero).
Bits 11-0 (Page 1)—The least significant bits of the Page 1 pointer are split into three subfields as shown in the following rows.	
11-5 (Page 1)	Reserved
4-3 (Page 1)	Transaction position (TP). This field is used with T-count to determine whether to send *all*, *first*, *middle*, or *last* with each outbound transaction payload. System software must initialize this field with the appropriate starting value. The host controller must correctly manage this state during the lifetime of the transfer. The bit encodings are: Value Meaning 00b All. The entire full-speed transaction data payload is in this transaction (that is, less than or equal to 188 bytes). 01b Begin. This is the first data payload for a full- speed that is greater than 188 bytes.transaction 10B Mid. This is the *middle* payload for a full-speed OUT transaction that is larger than 188 bytes. 11b End. This is the *last* payload for a full-speed OUT transaction that was larger than 188 bytes.
2-0 (Page 1)	Transaction count (T-Count). Software initializes this field with the number of OUT start-splits this transfer requires. Any value larger than 6 is undefined.

65.4.2.4.5 siTD Back Link Pointer

DWord 6 of a siTD is simply another schedule link pointer. This pointer is always zero, or references a siTD, and it cannot reference any other schedule data structure.

The following table describes the siTD back link pointer fields

.

Table 65-19. siTD Back Link Pointer

Bit	Description

Table continues on the next page...

Table 65-19. siTD Back Link Pointer (continued)

31-5	siTD Back Pointer. This field is a physical memory pointer to a siTD.
4-1	Reserved. This field is reserved for future use. It should be set to zero.
0	Terminate (T). 1 = siTD Back Pointer field is not valid. 0 = siTD Back Pointer field is valid.

65.4.2.5 Queue element transfer descriptor (qTD)

This data structure is only used with a queue head. It describes one or more USB transactions to transfer up to 20480 (5*4096) bytes.

The structure contains two structure pointers used for queue advancement, a DWord of transfer state, and a five-element array of data buffer pointers.

It is 32 bytes and must be physically contiguous.

The buffer associated with this transfer must be virtually contiguous. The buffer may start on any byte boundary; however, for optimal utilization of on-chip busses it is recommended to align the buffers on a 32-byte boundary. A separate buffer pointer list element must be used for each physical page in the buffer, regardless of whether the buffer is physically contiguous.

Host controller updates (host controller writes) to stand-alone qTDs only occur during transfer retirement. References in the following bit field definitions of updates to the qTD are to the qTD portion of a queue head.

The following table shows the queue element transfer descriptor data structure.

Table 65-20. Queue element transfer descriptor data structure

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Addr
Next qTD Pointer																											0				T	03-00
Alternate Next qTD Pointer																											0				T	07-04
dt	Total Bytes to Transfer															ioc	C_Page			Cerr		PID Code		Status								0B-08[1]
Buffer Pointer (page 0)																				Current Offset												0F-0C[1]
Buffer Pointer (page 1)																				Reserved												13-10
Buffer Pointer (page 2)																				Reserved												17-

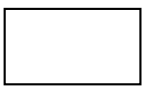
Table continues on the next page...

Table 65-20. Queue element transfer descriptor data structure (continued)

		14
Buffer Pointer (page 3)	Reserved	1B-18
Buffer Pointer (page 4)	Reserved	1F-1C

1. 08-0F: Transfer Results

Host Controller Read/Write



Host Controller Read Only

Queue Element Transfer Descriptors must be aligned on 32-byte boundaries.

65.4.2.5.1 Next qTD Pointer

The first DWord of an element transfer descriptor is a pointer to another transfer element descriptor.

The following table describes Next qTD pointer fields.

Table 65-21. qTD Next Element Transfer Pointer (DWord 0)

Bit	Description
31-5	Next Transfer Element Pointer. This field contains the physical memory address of the next qTD to be processed. The field corresponds to memory address signals[31:5], respectively.
4-1	Reserved
0	Terminate (T). 1= pointer is invalid. 0=Pointer is valid (points to a valid Transfer Element Descriptor). This bit indicates to the Host Controller that there are no more valid entries in the queue.

65.4.2.5.2 Alternate Next qTD Pointer

The second DWord of a queue element transfer descriptor is used to support hardware-only advance of the data stream to the next transfer descriptor on short packet. To be more explicit the host controller always uses this pointer when the current qTD is retired due to short packet.

The following table describes the TD Alternate Next Element Transfer Pointer field descriptions.

Table 65-22. TD Alternate Next Element Transfer Pointer (DWord 1)

Bit	Description

Table continues on the next page...

Table 65-22. TD Alternate Next Element Transfer Pointer (DWord 1) (continued)

31-5	Alternate Next Transfer Element Pointer. This field contains the physical memory address of the next qTD to be processed in the event that the current qTD execution encounters a short packet (for an IN transaction). The field corresponds to memory address signals [31:5], respectively.
4-1	Reserved
0	Terminate (T). 1= pointer is invalid. 0=Pointer is valid (points to a valid Transfer Element Descriptor). This bit indicates to the Host Controller that there are no more valid entries in the queue.

65.4.2.5.3 qTD Token

The third DWord of a queue element transfer descriptor contains most of the information the host controller requires to execute a USB transaction (the remaining endpoint-addressing information is specified in the queue head).

NOTE

The field descriptions forward reference fields defined in the queue head. Where necessary, these forward references are preceded with a QH notation.

The following table describes the TD Token fields.

Table 65-23. TD Token (DWord 2)

Bit	Description
31 Data Toggle	This is the data toggle sequence bit. The use of this bit depends on the setting of the *Data Toggle Control* bit in the queue head.
30-16 Total Bytes to Transfer	This field specifies the total number of bytes to be moved with this transfer descriptor. This field is decremented by the number of bytes actually moved during the transaction, only on the successful completion of the transaction. The maximum value software may store in this field is 5 * 4K (5000H). This is the maximum number of bytes 5 page pointers can access. If the value of this field is zero when the host controller fetches this transfer descriptor (and the active bit is set), the host controller executes a zero-length transaction and retires the transfer descriptor. It is not a requirement for OUT transfers that *Total Bytes To Transfer* be an even multiple of QHD.Maximum Packet Length. If software builds such a transfer descriptor for an OUT transfer, the last transaction is always less than QHD.Maximum Packet Length. Although it is possible to create a transfer up to 20K this assumes the 1st offset into the first page is 0. When the offset cannot be predetermined, crossing past the 5th page can be guaranteed by limiting the total bytes to 16K**. Therefore, the maximum recommended transfer is 16 K(4000H).
15 Interrupt On Complete (IOC)	If this bit is set to a one, it specifies that when this qTD is completed, the Host Controller should issue an interrupt at the next interrupt threshold.
14-12 Current Page (C_Page)	This field is used as an index into the qTD buffer pointer list. Valid values are in the range 0H to 4H. The host controller is not required to write this field back when the qTD is retired.
11-10 Error Counter (CERR)	This field is a 2-bit down counter that keeps track of the number of consecutive Errors detected while executing this qTD. If this field is programmed with a non-zero value during set-up, the Host Controller decrements the count and writes it back to the qTD if the transaction fails. If the counter counts from one to zero, the Host Controller marks the qTD

Table continues on the next page...

Table 65-23. TD Token (DWord 2) (continued)

<table>
<tr><th>Bit</th><th colspan="2">Description</th></tr>
<tr><td></td><td colspan="2">inactive, sets the Halted bit to a one, and error status bit for the error that caused CERR to decrement to zero. An interrupt is generated if the USB Error Interrupt Enable bit in the USBINTR register is set to a one. If HCD programs this field to zero during set-up, the Host Controller does not count errors for this qTD and there is no limit on the retries of this qTD. Note that write-backs of intermediate execution state are to the queue head overlay area, not the qTD.
Transaction Error - Decrement
Data Buffer Error - No Decrement[3]
Stalled - No Decrement[1]
Babble Detected - No Decrement[1]
No Error - No Decrement[2]</td></tr>
<tr><td></td><td>Error</td><td>Decrement Counter</td></tr>
<tr><td></td><td>1</td><td>Detection of Babble or Stall automatically halts the queue head. Thus, count is not decremented</td></tr>
<tr><td></td><td>2</td><td>If the EPS field indicates a HS device or the queue head is in the Asynchronous Schedule (and PIDCode indicates an IN or OUT) and a bus transaction completes and the host controller does not detect a transaction error, then the host controller should reset CERR to extend the total number of errors for this transaction. For example, CERR should be reset with maximum value (3) on each successful completion of a transaction. The host controller must never reset this field if the value at the start of the transaction is 00b.
See Split Transaction Interrupt for CERR adjustment rules when the EPS field indicates a FS or LS device and the queue head is in the Periodic Schedule. See Asynchronous - Do Complete Split for CERR adjustment rules when the EPS field indicates a FS or LS device, the queue head is in the Asynchronous schedule and the PIDCode indicates a SETUP.</td></tr>
<tr><td></td><td>3</td><td>Data buffer errors are host problems. They don't count against the device's retries.</td></tr>
<tr><td></td><td colspan="2">NOTE: Software must not program CERR to a value of zero when the EPS field is programmed with a value indicating a Full- or Low-speed device. This combination could result in undefined behavior.</td></tr>
<tr><td>9-8
PID Code</td><td colspan="2">This field is an encoding of the token, which should be used for transactions associated with this transfer descriptor. Encodings are:</td></tr>
<tr><td></td><td>00b</td><td>OUT Token generates token (E1H)</td></tr>
<tr><td></td><td>01b</td><td>IN Token generates token (69H)</td></tr>
<tr><td></td><td>10b</td><td>SETUP Token generates token (2DH) (undefined if endpoint is an interrupt, the queue head is non-zero) transfer type, for example, µFrame S-mask field in</td></tr>
<tr><td></td><td>11b</td><td>Reserved</td></tr>
<tr><td>7-0
Status</td><td colspan="2">This field is used by the Host Controller to communicate individual command execution states back to HCD. This field contains the status of the last transaction performed on this qTD. The bit encodings are:</td></tr>
<tr><td></td><td>Bit</td><td>Status Field Description</td></tr>
<tr><td></td><td>7</td><td>Active. Set to one by software to enable the execution of transactions by the Host Controller.</td></tr>
<tr><td></td><td>6</td><td>Halted. Set to one by the Host Controller during status updates to indicate that a serious error has occurred at the device/endpoint addressed by this qTD. This can be caused by babble, the error counter counting down to zero, or reception</td></tr>
</table>

Table continues on the next page...

Table 65-23. TD Token (DWord 2) (continued)

Bit	Description	
		of the STALL handshake from the device during a transaction. Any time that a transaction results in the Halted bit being set to a one, the Active bit is also set to zero.
	5	Data Buffer Error. Set to a one by the Host Controller during status update to indicate that the Host Controller is unable to keep up with the reception of incoming data (overrun) or is unable to supply data fast enough during transmission (under run). If an overrun condition occurs, the Host Controller forces a timeout condition on the USB, invalidating the transaction at the source. If the host controller sets this bit to a one, then it remains a one for the duration of the transfer.
	4	Babble Detected. Set to a one by the Host Controller during status update when" babble" is detected during the transaction. In addition to setting this bit, the Host Controller also sets the *Halted* bit to a one. Because "babble" is considered a fatal error for the transfer, setting the Halted bit to a one insures that no more transactions occur because of this descriptor.
	3	Transaction Error (XactErr). Set to a one by the Host Controller during status update in the case where the host did not receive a valid response from the device (Timeout, CRC, Bad PID, etc.). If the host controller sets this bit to a one, then it remains a one for the duration of the transfer.
	2	Missed Micro-Frame. This bit is ignored unless the *QH.EPS* field indicates a full- or low-speed endpoint and the queue head is in the periodic list. This bit is set when the host controller detected that a host-induced hold-off caused the host controller to miss a required complete-split transaction. If the host controller sets this bit to a one, then it remains a one for the duration of the transfer.
	1	Split Transaction State (SplitXstate). This bit is ignored by the host controller unless the *QH.EPS* field indicates a full- or low-speed endpoint. When a Full- or Low-speed device, the host controller uses this bit to track the state of the split-transaction. The functional requirements of the host controller for managing this state bit and the split transaction protocol depends on whether the endpoint is in the periodic or asynchronous schedule. The bit encodings are: Value Meaning 0b Do Start Split. This value directs the host controller to issue a Start split transaction to the endpoint. 1b Do Complete Split. This value directs the host controller to issue a Complete split transaction to the endpoint.
	0	Ping State (P)/ERR. If the *QH.EPS* field indicates a High-speed device and the *PID_Code* indicates an OUT endpoint, then this is the state bit for the Ping protocol. The bit encodings are: Value Meaning 0b Do OUT. This value directs the host controller to issue an OUT PID to the endpoint. 1b Do Ping. This value directs the host controller to issue a PING PID to the endpoint. If the *QH.EPS* field does not indicate a High-speed device, then this field is used as an error indicator bit. It is set to a one by the host controller whenever a periodic split-transaction receives an ERR handshake.

65.4.2.5.4 qTD Buffer Page Pointer List

The last five DWords of a queue element transfer descriptor is an array of physical memory address pointers. These pointers reference the individual pages of a data buffer.

System software initializes Current Offset field to the starting offset into the current page, where current page is selected through the value in the C_Page field.

The following table describes the qTD Buffer Pointer(s) (DWords 3-7) fields.

Table 65-24. qTD Buffer Pointer(s) (DWords 3-7)

Bit	Description
31-12	Buffer Pointer List. Each element in the list is a 4 K page aligned physical memory address. The lower 12 bits in each pointer are reserved (except for the first one), as each memory pointer must reference the start of a 4 K page. The field C_Page specifies the current active pointer. When the transfer element descriptor is fetched, the starting buffer address is selected using C_Page (similar to an array index to select an array element). If a transaction spans a 4K buffer boundary, the host controller must detect the page-span boundary in the data stream, increment C_Page and advance to the next buffer pointer in the list, and conclude the transaction through the new buffer pointer.
11-0	Current Offset (Reserved). This field is reserved in all pointers except the first one (for example Page 0). The host controller should ignore all reserved bits. For the page 0 current offset interpretation, this field is the byte offset into the current page (as selected by C_Page). The host controller is not required to write this field back when the qTD is retired. Software should ensure the Reserved fields are initialized to zero.

65.4.2.6 Queue Head

The following table shows the queue head structure layout.

Table 65-25. Queue Head Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Addr
Queue Head Horizontal Link Pointer																											0		Typ		T	03-00
RL				C	Maximum Packet Length											H	dtc	EP		EndPt				I	Device Address							07-04[1]
Mult		Port Number[2]							Hub Addr[2]							μFrame C-mask[2]								μFrame S-mask								0B-08[1]
Current qTD Pointer																											0					0F-0C
Next qTD Pointer																											0				T	13-10[3]
Alternate Next qTD pointer																											NakCnt				T	17-14[4]
dt	Total Bytes to Transfer															ioc	C_Page			Cerr		PID Code		Status								1B-18
Buffer Pointer (Page 0)																				Current Offset												1F-1C
Buffer Pointer (Page 1)																				Reserved				C-prog-mask[2]								23-20

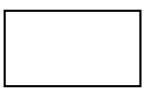
Table continues on the next page...

Table 65-25. Queue Head Structure Layout (continued)

Buffer Pointer (Page 2)	S-bytes[2]	FrameTag[2]	27-24[4]
Buffer Pointer (Page 3)	Reserved		2B-28
Buffer Pointer (Page 4)	Reserved		2F-2C[3]

1. 04-0B: Static endpoint state.
2. These fields are used exclusively to support split transactions to USB 2.0 hubs
3. 10-2F: Transfer overlay.
4. 14-27: Transfer results.

Host Controller Read/Write



Host Controller Read Only

65.4.2.6.1 Queue Head Horizontal Link Pointer

The first DWord of a Queue Head contains a link pointer to the next data object to be processed after any required processing in this queue has been completed, as well as the control bits defined below.

This pointer may reference a queue head or one of the isochronous transfer descriptors. It must not reference a queue element transfer descriptor.

The following table describes the Queue head DWord 0 fields.

Table 65-26. Queue Head DWord 0

Bit	Description
31-5	Queue Head Horizontal Link Pointer (QHLP). This field contains the address of the next data object to be processed in the horizontal list and corresponds to memory address signals [31:5], respectively.
4-3	Reserved
2-1	QH/(s)iTD Select (Typ). This field indicates to the hardware whether the item referenced by the link pointer is an iTD, siTD or a QH. This allows the Host Controller to perform the proper type of processing on the item after it is fetched. Value encodings are: Value Meaning 00b iTD (isochronous transfer descriptor) 01b QH (queue head) 10b siTD (split transaction isochronous transfer descriptor) 11b FSTN (frame span traversal node)
0	Terminate (T). 1=Last QH (pointer is invalid). 0=Pointer is valid. If the queue head is in the context of the periodic list, a one bit in this field indicates to the host controller that this is the end of the periodic list. This bit is ignored by the host controller when the queue head is in the Asynchronous schedule. Software must ensure that queue heads reachable by the host controller always have valid horizontal link pointers.

65.4.2.6.2 Queue Head Endpoint Capabilities/Characteristics

The second and third DWords of a Queue Head specifies static information about the endpoint. This information does not change over the lifetime of the endpoint.

There are three types of information in this region:

- Endpoint Characteristics. These are the USB endpoint characteristics including addressing, maximum packet size, and endpoint speed.
- Endpoint Capabilities. These are adjustable parameters of the endpoint. They effect how the endpoint data stream is managed by the host controller.
- Split Transaction Characteristics. This data structure is used to manage full- and low-speed data streams for bulk, control, and interrupt via split transactions to USB2.0 Hub Transaction Translator. There are additional fields used for addressing the hub and scheduling the protocol transactions (for periodic).

The host controller must not modify the bits in this region.

The following table describes the Endpoint characteristics: Queue head DWord 1 fields.

Table 65-27. Endpoint Characteristics: Queue Head DWord 1

Bit	Description	
31-28	Nak Count Reload (RL). This field contains a value, which is used by the host controller to reload the Nak Counter field.	
27	Control Endpoint Flag (C). If the *QH.EPS* field indicates the endpoint is not a high-speed device, and the endpoint is a control endpoint, then software must set this bit to a one. Otherwise, it should always set this bit to zero.	
26-16	Maximum Packet Length. This directly corresponds to the maximum packet size of the associated endpoint (*wMaxPacketSize*). The maximum value this field may contain is 0x400 (1024).	
15	Head of Reclamation List Flag (H). This bit is set by System Software to mark a queue head as being the head of the reclamation list.	
14	Data Toggle Control (DTC). This bit specifies where the host controller should get the initial data toggle on an overlay transition. 0b Ignore DT bit from incoming qTD. Host controller preserves DT bit in the queue head. 1b Initial data toggle comes from incoming qTD DT bit. Host controller replaces DT bit in the queue head from the DT bit in the qTD.	
13-12	Endpoint Speed (EPS). This is the speed of the associated endpoint. Bit combinations are:	
	Value	Meaning
	00b	Full-Speed (12 Mbits/sec)
	01b	Low-Speed (1.5 Mbits/sec)
	10b	High-Speed (480 Mbits/sec)
	11b	Reserved
	This field must not be modified by the host controller.	
11-8	Endpoint Number (Endpt). This 4-bit field selects the particular endpoint number on the device serving as the data source or sink.	

Table continues on the next page...

Table 65-27. Endpoint Characteristics: Queue Head DWord 1 (continued)

7	Inactivate on Next Transaction (I). This bit is used by system software to request that the host controller set the Active bit to zero. See Rebalancing the periodic schedule , for full operational details. This field is only valid when the queue head is in the Periodic Schedule and the *EPS* field indicates a Full or Low-speed endpoint. Setting this bit to one when the queue head is in the Asynchronous Schedule or the *EPS* field indicates a high-speed device yields undefined results.
6-0	Device Address. This field selects the specific device serving as the data source or sink.

The table below describes the Endpoint capabilities: Queue head DWord 2 field descriptions.

Table 65-28. Endpoint Capabilities: Queue Head DWord 2

Bit	Description
31-30	High-Bandwidth Pipe Multiplier (Mult). This field is a multiplier used to key the host controller as the number of successive packets the host controller may submit to the endpoint in the current execution. The host controller makes the simplifying assumption that software properly initializes this field (regardless of location of queue head in the schedules or other run time parameters). The valid values are: Value Meaning 00b Reserved. A zero in this field yields undefined results. 01b One transaction to be issued for this endpoint per micro-frame 10b Two transactions to be issued for this endpoint per micro-frame 11b Three transactions to be issued for this endpoint per micro-frame
29-23	Port Number. This field is ignored by the host controller unless the *EPS* field indicates a full- or low-speed device. The value is the port number identifier on the USB 2.0 Hub (for hub at device address *Hub Addr* below), below which the full- or low-speed device associated with this endpoint is attached. This information is used in the split-transaction protocol.
22-16	Hub Addr. This field is ignored by the host controller unless the *EPS* field indicates a full-or low-speed device. The value is the USB device address of the USB 2.0 Hub below which the full- or low-speed device associated with this endpoint is attached. This field is used in the split-transaction protocol.
15-8	Split Completion Mask (µFrame C-Mask). This field is ignored by the host controller unless the *EPS* field indicates this device is a low- or full-speed device and this queue head is in the periodic list. This field (along with the *Active* and *SplitX-state* fields) is used to determine during which micro-frames the host controller should execute a complete-split transaction. When the criteria for using this field are met, a zero value in this field has undefined behavior. This field is used by the host controller to match against the three low-order bits of the FRINDEX register. If the FRINDEX register bits decode to a position where the µ*Frame C- Mask* field is a one, then this queue head is a candidate for transaction execution. There may be more than one bit in this mask set.
7-0	Interrupt Schedule Mask (µFrame S-mask). This field is used for all endpoint speeds. Software should set this field to a zero when the queue head is on the asynchronous schedule. A non-zero value in this field indicates an interrupt endpoint. The host controller uses the value of the three low-order bits of the FRINDEX register as an index into a bit position in this bit vector. If the µ*Frame S-mask* field has a one at the indexed bit position then this queue head is a candidate for transaction execution. If the *EPS* field indicates the endpoint is a high-speed endpoint, then the transaction executed is determined by the *PID_Code* field contained in the execution area. This field is also used to support split transaction types: Interrupt (IN/OUT). This condition is true when this field is non-zero and the *EPS* field indicates this is either a full- or low-speed device. A zero value in this field, in combination with existing in the periodic frame list has undefined results.

65.4.2.6.3 Transfer Overlay-Queue Head

The nine DWords in this area represent a transaction working space for the host controller. The general operational model is that the host controller can detect whether the overlay area contains a description of an active transfer. If it does not contain an active transfer, then it follows the Queue Head Horizontal Link Pointer to the next queue head. The host controller will never follow the Next Transfer Queue Element or Alternate Queue Element pointers unless it is actively attempting to advance the queue. For the duration of the transfer, the host controller keeps the incremental status of the transfer in the overlay area. When the transfer is complete, the results are written back to the original queue element.

The DWord3 of a Queue Head contains a pointer to the source qTD currently associated with the overlay. The host controller uses this pointer to write back the overlay area into the source qTD after the transfer is complete.

The following table describes the current qTD link pointer field descriptions.

Table 65-29. Current qTD Link Pointer

Bit	Description
31-5	Current Element Transaction Descriptor Link Pointer. This field contains the address Of the current transaction being processed in this queue and corresponds to memory address signals [31:5], respectively.
4-0	Reserved (R). These bits are ignored by the host controller when using the value as an address to write data. The actual value may vary depending on the usage.

The DWords 4-11 of a queue head are the transaction overlay area. This area has the same base structure as a Queue Element Transfer Descriptor. The queue head utilizes the reserved fields of the page pointers to implement tracking the state of split transactions.

This area is characterized as an overlay because when the queue is advanced to the next queue element, the source queue element is merged onto this area. This area serves as execution cache for the transfer.

The table below describes the Host-controller rules for bits in overlay.

Table 65-30. Host-Controller Rules for Bits in Overlay (DWords 5, 6, 8 and 9)

DWord	Bit	Description
5	4-1	Nak Counter (NakCnt)µRW. This field is a counter the host controller decrements whenever a transaction for the endpoint associated with this queue head results in a Nak or Nyet response. This counter is reloaded from *RL* before a transaction is executed during the first pass of the reclamation list (relative to an Asynchronous List Restart condition). It is also loaded from RL during an overlay.
6	31	Data Toggle. The *Data Toggle Control* controls whether the host controller preserves this bit when an overlay operation is performed.

Table continues on the next page...

Table 65-30. Host-Controller Rules for Bits in Overlay (DWords 5, 6, 8 and 9) (continued)

6	15	Interrupt On Complete (IOC). The IOC control bit is always inherited from the source qTD when the overlay operation is performed.
6	11-10	Error Counter (C_ERR). This two-bit field is copied from the qTD during the overlay and written back during queue advancement.
6	0	Ping State (P)/ERR. If the *EPS* field indicates a high-speed endpoint, then this field should be preserved during the overlay operation.
8	7-0	Split-transaction Complete-split Progress (C-prog-mask). This field is initialized to zero during any overlay. This field is used to track the progress of an interrupt split-transaction.
9	4-0	Split-transaction Frame Tag (Frame Tag). This field is initialized to zero during any overlay. This field is used to track the progress of an interrupt split-transaction.
9	11-5	S-bytes. Software must ensure that the *S-bytes* field in a *qTD* is zero before activating the *qTD*. This field is used to keep track of the number of bytes sent or received during an IN or OUT split transaction.

65.4.2.7 Periodic Frame Span Traversal Node (FSTN)

This data structure is to be used only for managing Full- and Low-speed transactions that span a Host-frame boundary.

See Host Controller Operational Model for FSTNs for full operational details. Software must not use an FSTN in the Asynchronous Schedule. An FSTN in the Asynchronous schedule results in undefined behavior. Software must not use the FSTN feature with a host controller whose USB_HCIVERSION register indicates a revision implementation below 0096h. FSTNs are not defined for implementations before 0.96 and their use yields undefined results.

Table 65-31. Frame Span Traversal Node Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Addr
Normal Path Link Pointer																											0		Typ		T	03-00
Back Path Link Pointer																											0		Typ[1]		T	07-04

1. Must be set to indicate a queue head

Host Controller Read/Write Host Controller Read Only

65.4.2.7.1 FSTN Normal Path Pointer

The first DWord of an FSTN contains a link pointer to the next schedule object. This object can be of any valid periodic schedule data type.

The following table describes the FSTN normal path pointer fields.

Table 65-32. FSTN Normal Path Pointer Field Descriptions

Bit	Description
31-5	Normal Path Link Pointer (NPLP). This field contains the address of the next data object to be processed in the periodic list and corresponds to memory address signals [31:5], respectively.
4-3	Reserved
2-1	QH/(s)iTD/FSTN Select (Typ). This field indicates to the Host Controller whether the item referenced is a iTD/siTD, a QH or an FSTN. This allows the Host Controller to perform the proper type of processing on the item after it is fetched. Value encodings are: Value Meaning 00b iTD (isochronous transfer descriptor) 01b QH (queue head) 10b siTD (split transaction isochronous transfer descriptor) 11b FSTN (Frame Span Traversal Node)
0	Terminate (T). 1=Link Pointer field is not valid. 0=Link Pointer is valid.

65.4.2.7.2 FSTN Back Path Link Pointer

The second DWord of an FTSN node contains a link pointer to a queue head.

If the T-bit in this pointer is zero, then this FSTN is a Save-Place indicator. Its Typ field must be set by software to indicate the target data structure is a queue head. If the T-bit in this pointer is set to one, then this FSTN is the Restore indicator. When the T-bit is one, the host controller ignores the Typ field.

The following table describes the FSTN back path link pointer fields.

Table 65-33. FSTN Back Path Link Pointer Field Descriptions

Bit	Description
31-5	Back Path Link Pointer (BPLP). This field contains the address of a Queue Head. This field corresponds to memory address signals [31:5], respectively.
4-3	Reserved
2-1	Typ. Software must ensure this field is set to indicate the target data structure is a Queue Head. Any other value in this field yields undefined results.
0	Terminate (T). 1=Link Pointer field is not valid (that is the host controller must not use bits [31:5] as a valid memory address). This value also indicates that this FSTN is a Restore indicator. 0=Link Pointer is valid (that is the host controller may use bits [31:5] (in combination with the CTRLDSSEGMENT register if applicable) as a valid memory address). This value also indicates that this FSTN is a Save-Place indicator.

65.4.3 Host Operational Model

The general operational model is for the enhanced interface host controller hardware and enhanced interface host controller driver (generally referred to as system software).

Each significant operational feature of the EHCI host controller is discussed in a separate section. Each section presents the operational model requirements for the host controller hardware. Where appropriate, recommended system software operational models for features are also presented.

65.4.3.1 Host Controller Initialization

After initial power-on or HCReset (hardware or through HCReset bit in the USB_USBCMD register), all of the operational registers are at their default values. After a hardware reset, only the operational registers not contained in the Auxiliary power well are at their default values.

The following table describes the default values of operational registers.

.

Table 65-34. Default Values of Operational Register Space

Operational Register	Default Value (after Reset)
USB_USBCMD	00080000h (00080B00h, if *Asynchronous Schedule Park Capability is one*)
USB_USBSTS	00001000h
USB_USBINTR	00000000h
USB_FRINDEX	00000000h
USB_CTRLDSSEGMENT	00000000h
USB_PERIODICLISTBASE	Undefined
USB_ASYNCLISTADDR	Undefined
USB_CONFIGFLAG	00000000h
USB_PORTSC1	00002000h (w/*PPC* set to one); 00003000h (w/*PPC* set to zero)

To initialize the host controller, software should perform the following steps:

- Write the appropriate value to the USB_USBINTR register to enable the appropriate interrupts.
- Write the base address of the Periodic Frame List to the USB_PERIODICLIST BASE register. If no work items are in the periodic schedule, all elements of the Periodic Frame List should have their T-Bits set to one.
- Write the USB_USBCMD register to set the desired interrupt threshold, frame list size (if applicable) and turn the host controller ON through setting the Run/Stop bit.

At this point, the host controller is up and running and the port registers begin reporting device connects, and so on. System software can enumerate a port through the reset process (where the port is in the enabled state). At this point, the port is active with SOFs occurring down the enabled ports, but the schedules have not enabled. To communicate

with devices through the asynchronous schedule, system software must write the USB_ASYNCLISTADDR register with the address of a control or bulk queue head. Software must then enable the asynchronous schedule by writing one to the Asynchronous Schedule Enable bit in the USB_USBCMD register. To communicate with devices through the periodic schedule, system software must enable the periodic schedule by writing one to the Periodic Schedule Enable bit in the USB_USBCMD register.

NOTE

The schedules can be turned on before the first port is reset (and enabled).

When the USB_USBCMD register is written, system software must ensure the appropriate bits are preserved, depending on the intended operation.

65.4.3.2 Port Routing and Control

The EHCI specification defines that a USB 2.0 Host controller is comprised of one high-speed host controller, which implements the EHCI programming interface and 0 to N USB 1.1 companion host controllers.

Companion host controllers (cHCs) may be implementations of either Universal or Open host controller specifications. This configuration is used to deliver the required full USB 2.0-defined port capability; for example, Low-, Full-, and High-speed capability for every port.

NOTE

The USB controllers on i.MX parts do not require nor support companion controllers to support Full and Low Speed device. Full and Low Speed devices are supported within the USB controller by emulating the functionality of a high-speed HUB. Therefore, no port routing is present in the controller. Please refer to Embedded Transaction Translator Function for detail!

The following figure illustrates a simple block diagram of the port routing logic and its relationship to the high-speed and companion host controllers within a USB 2.0 host controller.

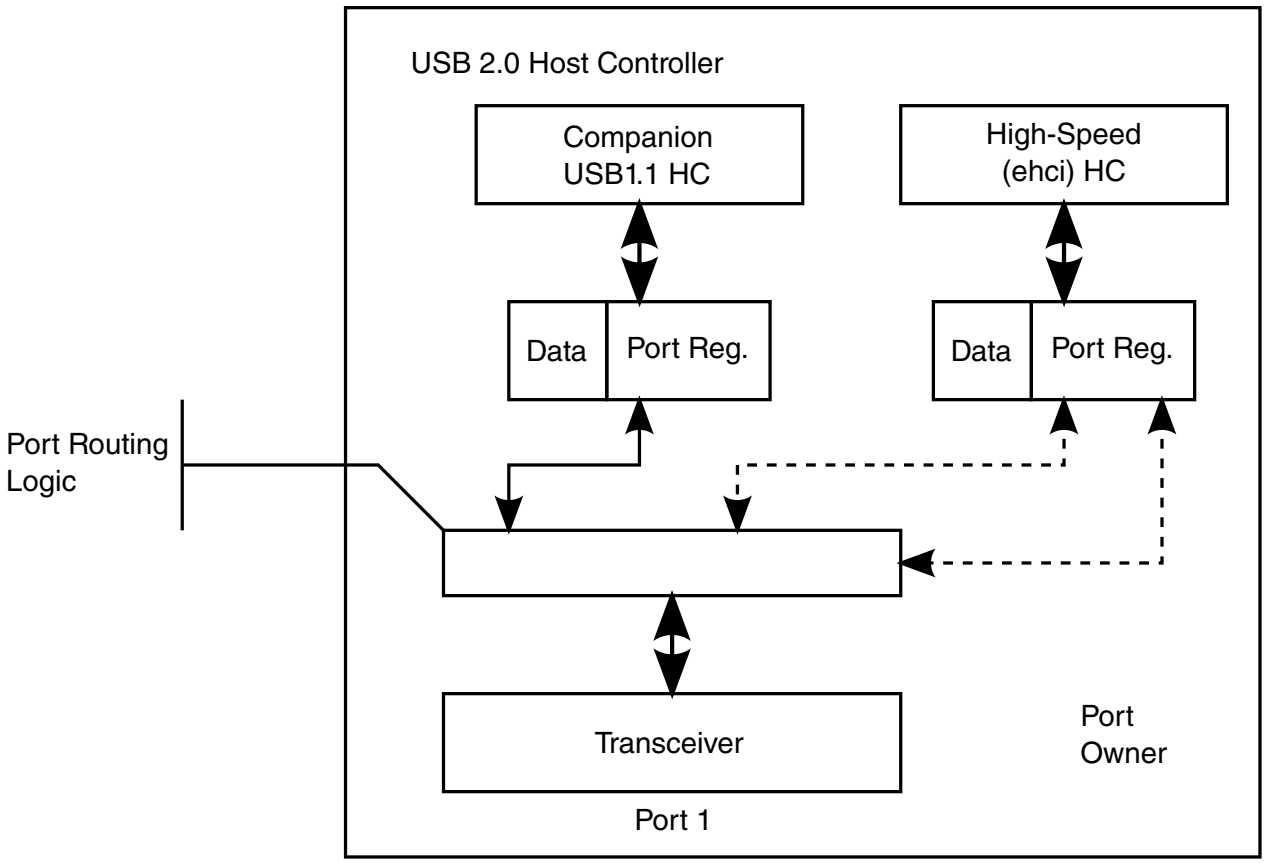


Figure 65-5. Example USB 2.0 Host Controller Port Routing Block Diagram

There exists one transceiver per physical port and each host controller block has its own port status and control registers. The EHCI controller has port status and control registers for every port. Each companion host controller has only the port control and status registers it is required to operate. Either the EHCI host controller or one companion host controller controls each transceiver. Routing logic lies between the transceiver, the port status and control registers.[1]

The port routing logic is controlled from signals originating in the EHCI host controller. The EHCI host controller has a global routing policy control field and per-port ownership control fields. The Configured Flag (CF) bit is the global routing policy control. At power-on or reset, the default routing policy is to the companion controllers (if they exist). If the system does not include a driver for the EHCI host controller and the host controller includes Companion Controllers, then the ports still work in Full- and Low-speed mode (assuming the system includes a driver for the companion controllers). In general, when the EHCI owns the ports, the companion host controllers' port registers do not see a connect indication from the transceiver. Similarly, when a companion host controller owns a port, the EHCI controller's port registers do not see a connect indication

1. The routing logic should not be implemented in the 480 MHz clock domain of the transceiver.

from the transceiver. The details on the rules for the port routing logic are described in the following sections. The USB 2.0 host controller must be implemented as a multi-function PCI device if the implementation includes companion controllers. The companion host controllers' function numbers must be less than the EHCI host controller function number. The EHCI host controller must be a larger function number with respect to the companion host controllers associated with this EHCI host controller. If a PCI device implementation contains only an EHCI controller (that is no companion controllers or other PCI functions), then the EHCI host controller must be function zero, in accordance with the PCI Specification. The N_CC field in the Structural Parameter register (HCSPARAMS) indicates whether the controller implementation includes companion host controllers. When N_CC has a non-zero value there exists companion host controllers. If N_CC has a value of zero, then the host controller implementation does not include companion host controllers. If the host controller root ports are exposed to attachment of full- or low-speed devices, the ports always fails the high-speed chirp during reset and the ports are not enabled. System software can notify the user of the illegal condition. This type of implementation requires a USB 2.0 hub be connected to a root port to provide full and low-speed device connectivity.

System software uses information in the host controller capability registers to determine how the ports are routed to the companion host controllers. See Host Controller Structural Parameters (USBC_*n*_HCSPARAMS).

65.4.3.2.1 Port Routing Control through EHCI Configured (CF) Bit

Each port in the USB 2.0 host controller are routed either to a single companion host controller or to the EHCI host controller.

The port routing logic is controlled by two mechanisms in the EHCI HC: a host controller global flag and per-port control. The Configured Flag (CF) bit, is used to globally set the policy of the routing logic. Each port register has a Port Owner control bit which allows the EHCI Driver to explicitly control the routing of individual ports. Whenever the CF bit transitions from zero to one (this transition is only available under program control) the port routing unconditionally routes all of the port registers to the EHCI HC (all Port Owner bits go to zero). While the CF-bit is one, the EHCI Driver controls individual ports' routing through the Port Owner control bit. Likewise, whenever the CF bit transitions from one to zero (as a result of Aux power application, HCRESET, or software writing zero to CF-bit), the port routing unconditionally routes all of the port registers to the appropriate companion HC. The default value for the EHCI HC's CF bit (after Aux power application or HCRESET) is zero.

The *view* of the port depends on the current owner. A Universal or Open companion host controller will see port register bits consistent with the appropriate specification. Port bit definitions that are required for EHCI host controllers are not visible to companion host controllers.

The following table summarizes the default routing for all the ports, based on the value of the EHCI HC's CF bit.

Table 65-35. Default Port Routing Depending on EHCI HC CF Bit

HS CF Bit	Default Port Ownership	Explanation
0B	Companion HCs	The companion host controllers own the ports and only Full- and Low-speed devices are supported in the system. The exact port assignments are implementation dependent. The ports behave only as Full- and Low-speed ports in this configuration
1B	EHCI HC	The EHCI host controller has default ownership over all of the ports. The routing logic inhibits device connect events from reaching the companion HCs' port status and control registers when the port owner is the EHCI HC.The EHCI HC has access to the additional port status and control bits defined in this specification (see Port Status & Control (USBC_*n*_PORTSC1)). The EHCI HC can temporarily release control of the port to a companion HC by setting the *PortOwner* bit in the PORTSC1 register to one.

65.4.3.2.2 Port Routing Control through PortOwner and Disconnect Event

Manipulating the port routing through the CF-bit is an extreme process and not intended to be used during normal operation.

The normal mode of port ownership transferal is on the granularity of individual ports using the Port Owner bit in the EHCI HC's USB_PORTSC1 register (for hand-offs from EHCI to companion host controllers). Individual port ownership is returned to the EHCI controller when the port registers a device disconnect. When the disconnect is detected, the port routing logic immediately returns the port ownership to the EHCI controller. The companion host controller port register detects the device disconnect and operates normally.

Under normal operating conditions (assuming all HC drivers loaded and operational and the EHCI *CF-bit* is set to one), the typical port enumeration sequence proceeds as illustrated below:

- Initial condition is that EHCI is port owner. A device is connected causing the port to detect a connect, set the port connect change bit and issue a port-change interrupt (if enabled).
- EHCI Driver identifies the port with the new connect change bit asserted and sends a change report to the hub driver. Hub driver issues a GetPortStatus() request and

identifies the connect change. It then issues a request to clear the connect change, followed by a request to reset and enable the port.

- When the EHCI Driver receives the request to reset and enable the port, it first checks the value reported by the LineStatus bits in the USB_PORTSC1 register. If they indicate the attached device is a full-speed device (for example, D+ is asserted), then the EHCI Driver sets the PortReset control bit to one (and sets the PortEnable bit to zero) which begins the reset-process. Software times the duration of the reset, then terminates reset signaling by writing zero to the port reset bit. The reset process is actually complete when software reads zero in the PortReset bit. The EHCI Driver checks the PortOwner bit in the USB_PORTSC1 register. If set to one, the connected device is a high-speed device and EHCI Driver (root hub emulator) issues a change report to the hub driver and the hub driver continues to enumerate the attached device.
- At the time the EHCI Driver receives the port reset and enable request the LineStatus bits might indicate a low-speed device. Additionally, when the port reset process is complete, the PortEnable field may indicate that a full-speed device is attached. In either case the EHCI driver sets the PortOwner bit in the USB_PORTSC1 register to one to release port ownership to a companion host controller.
- When the EHCI Driver sets PortOwner bit to one, the port routing logic makes the connection state of the transceiver available to the companion host controller port register and removes the connection state from the EHCI HC port. The EHCI USB_PORTSC1 register observes and reports a disconnect event through the disconnect change bit. The EHCI Driver detects the connection status change (either by polling or by port change interrupt) and then sends a change report to the hub driver. When the hub driver requests that port-state, the EHCI Driver responds with a reset complete change set to one, a connect change set to one and a connect status set to zero. This information is derived directly from the EHCI port register. This allows the hub driver to assume the device was disconnected during reset. It acknowledges the change bits and wait for the next change event. While the EHCI controller does not own the port, it simply remains in a state where the port reports no device connected. The device-connect evaluation circuitry of the companion HC activates and detects the device, the companion Driver detects the connection and enumerates the port.

When a port is routed to a companion HC, it remains under the control of the companion HC until the device is disconnected from the root port (ignoring for now the scenario where EHCI's CF-bit transitions from1b to 0b). When a disconnect occurs, the disconnect event is detected by both the companion HC port control and the EHCI port ownership control. On the event, the port ownership is returned immediately to the EHCI controller. The companion HC stack detects the disconnect and acknowledges as it would in an ordinary standalone implementation. Subsequent connects is detected by the EHCI port register and the process repeats.

65.4.3.2.3 Example Port Routing State Machine

The following figure illustrates an example of how the port ownership should be managed. The following sections describe the entry conditions to each state.

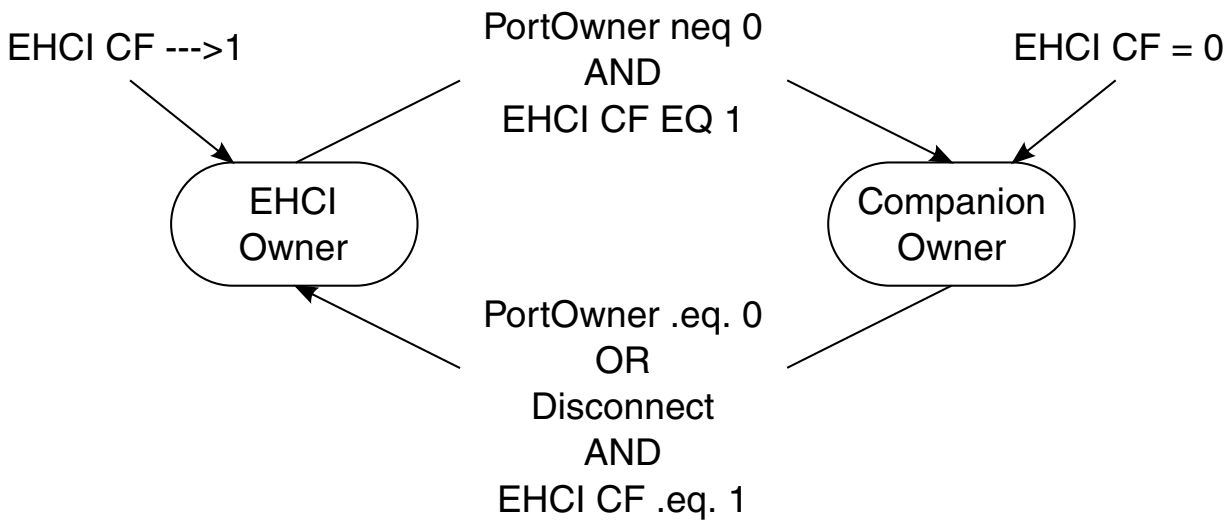


Figure 65-6. Port Owner Handoff State Machine

65.4.3.2.3.1 EHCI HC Owner

Entry to this state occurs when one of the following events occur:

- When the EHCI HC's Configure Flag (CF) bit in the USB_CONFIGFLAG register transitions from zero to one. This signals the fact that the system has a host controller driver for the EHCI HC and that all ports in the USB 2.0 host controller must default route to the EHCI controller.
- When the port is owned by a companion HC and the device is disconnected from the port. The EHCI port routing control logic is notified of the disconnect, and returns port routing to the EHCI controller. The connection state of the companion HC goes immediately to the disconnected state (with appropriate side effect to connect change, enable and enable change). The companion HC driver acknowledges the disconnect by setting the connect status change bit to zero. This allows the companion HC's driver to interact with the port completely through the disconnect process.
- When system software writes zero to the PortOwner bit in the USB_PORTSC1 register. This allows software to take ownership of a port from a companion host controller. When this occurs, the routing logic to the companion HC effectively signals a disconnect to the companion HC's port status and control register.

65.4.3.2.3.2 Companion HC Owner

Entry to this state occurs whenever one of the following events occur:

- When the PortOwner field transitions from zero to one.
- When the HS-mode HC's Configure Flag (CF) is equal to zero.

On entry to this state, the routing logic allows the companion HC port register to detect a device connect. Normal port enumeration proceeds.

65.4.3.2.4 Port Power

The Port Power Control (PPC) bit in the USB_HCSPARAMS register indicates whether the USB 2.0 host controller has port power control (see Host Controller Structural Parameters (USBC_*n*_HCSPARAMS)).

When this bit is zero, then the host controller does not support software control of port power switches. When in this configuration, the port power is always available and the companion host controllers must implement functionality consistent with port power always on. When the *PPC* bit is one, then the host controller implementation includes port power switches. Each available switch has an output enable, which is referred to in this discussion as PortPowerOutputEnable (PPE). PPE is controlled based on the state of the combination bits PPC bit, EHCI Configured (CF)-bit and individual Port Power (PP) bits.

The following table describes the summary behavioral model.

Table 65-36. Port Power Enable Control Rules

CF	CHC[1] (PP)	EHC[2] (PP)	Owner	PPE[3]	Description
0	0	X	CHC	0	When the EHCI controller is not configured, the port is owned by the companion host controller. When the companion HC's port power select is off, then the port power is off.
0	1	X	CHC	1	Similar to previous entry. When the companion HC's port power select is on, then the port power is on.
1	0	0	CHC	0	Port owner has port power turned off, the power to port is off.
1	0	0	EHC	0	Port owner has port power turned off, the power to port is off.
1	0	1	EHC	1	Port owner has port power on, so power to port is on.
1	0	1	CHC	1	If either HC has port power turned on, the power to the port is on.

Table continues on the next page...

Table 65-36. Port Power Enable Control Rules (continued)

1	1	0	EHC	1	If either HC has port power turned on, the power to the port is on.
1	1	0	CHC	1	Port owner has port power on, so power to port is on.
1	1	1	CHC	1	Port owner has port power on, so power to port is on.
1	1	1	EHC	1	Port owner has port power on, so power to port is on.

1. CHC (Companion Host Controller).
2. EHC (EHCI Host Controller).
3. PPE (Port Power Enable). This bit actually turns on the port power switch (if one exists).

65.4.3.2.5 Port Reporting Over-Current

Host controllers are by definition power providers on USB. Whether the ports are considered high- or low-powered is a platform implementation issue. Each EHCI USB_PORTSC1 register has an over-current status and over-current change bit.

The functionality of these bits are specified in the USB Specification Revision 2.0.

The over current detection and limiting logic usually resides outside the host controller logic. This logic may be associated with one or more ports. When this logic detects an over-current condition it is made available to both the companion and EHCI ports. The effect of an over-current status on a companion host controller port is beyond the scope of this document.

The over-current condition effects the following bits in the USB_PORTSC1 register on the EHCI port:

- Over-current Active bits are set to one. When the over-current condition goes away, the Over-current Active bit transitions from one to zero.
- Over-current Change bits are set to one. On every transition of the Over-current Active bit the host controller sets the Over-current Change bit to one. Software sets the Over-current Change bit to zero by writing one to this bit.
- Port Enabled/Disabled bit is set to zero. When this change bit gets set to one, then the Port Change Detect bit in the USB_USBSTS register is set to one.
- Port Power (PP) bits may optionally be set to zero. There is no requirement in USB that a power provider shut off power in an over current condition. It is sufficient to limit the current and leave power applied. When the Over-current Change bit transitions from zero to one, the host controller also sets the Port Change Detect bit in the USB_USBSTS register to one. In addition, if the Port Change Interrupt Enable bit in the USB_USBINTR register is one, then the host controller issues an interrupt to the system. Refer to Table 65-37 for summary behavior for over-current detection

when the host controller is halted (suspended from a device component point of view).

65.4.3.3 Suspend/Resume-Host Operational Model

The EHCI host controller provides an equivalent suspend and resume model as that defined for individual ports in a USB 2.0 Hub.

Control mechanisms are provided to allow system software to suspend and resume individual ports. The mechanisms allow the individual ports to be resumed completely through software initiation. Other control mechanisms are provided to parameterize the host controller's response (or sensitivity) to external resume events. In this discussion, host-initiated, or software initiated resumes are called Resume Events/Actions. Bus-initiated resume events are called wake-up events. The classes of wake-up events are:

- Remote-wake-up enabled device asserts resume signaling. In similar kind to USB 2.0 Hubs, EHCI controllers must always respond to explicit device resume signaling and wake-up the system (if necessary).
- Port connect and disconnect and over-current events. Sensitivity to these events can be turned on or off by using the per-port control bits in the USB_PORTSC1 registers.

Selective suspend is a feature supported by every USB_PORTSC1 register. It is used to place specific ports into a suspend mode. This feature is used as a functional component for implementing the appropriate power management policy implemented in a particular operating system. When system software intends to suspend the entire bus, it should selectively suspend all enabled ports, then shut off the host controller by setting the Run/Stop bit in the USB_USBCMD register to zero. The EHCI sub-block can then be placed into a lower device state through the PCI power management interface (see Appendix A, Enhanced Host Controller Interface Specification for Universal Serial Bus, Revision 0.95, November 2000, Intel Corporation. http://www.intel.com).

When a wake event occurs, the system resumes operation and system software eventually set the Run/Stop bit to one and resume the suspended ports. Software must not set the Run/Stop bit to one until it is confirmed that the clock to the host controller is stable. This is usually confirmed in a system implementation in that all of the clocks in the system are stable before the ARM platform is restarted. So, by definition, if software is running, clocks in the system are stable and the Run/Stop bit in the USB_USBCMD register can be set to one. Minimum system software delays are also defined in the PCI Power Management Specification. Refer to PCI Power Management Specification for more information.

65.4.3.3.1 Port Suspend/Resume

System software places individual ports into suspend mode by writing one into the appropriate USB_PORTSC1 Suspend bit. Software must only set the Suspend bit when the port is in the enabled state (Port Enabled bit is one) and the EHCI is the port owner (PortOwner bit is zero).

The host controller may evaluate the Suspend bit immediately or wait until a micro-frame or frame boundary occurs. If evaluated immediately, the port is not suspended until the current transaction (if one is executing) completes. Therefore, there may be several micro-frames of activity on the port until the host controller evaluates the Suspend bit. The host controller must evaluate the Suspend bit at least every frame boundary.

System software can initiate a resume on a selectively suspended port by writing one to the Force Port Resume bit. Software should not attempt to resume a port unless the port reports that it is in the suspended state (see Port Status & Control (USBC_*n*_PORTSC1)). If system software sets Force Port Resume bit to one when the port is not in the suspended state, the resulting behavior is undefined. In order to assure proper USB device operation, software must wait for at least 10 ms after a port indicates that it is suspended (Suspend bit is one) before initiating a port resume through the Force Port Resume bit. When Force Port Resume bit is one, the host controller sends resume signaling down the port. System software times the duration of the resume (nominally 20 ms) then sets the Force Port Resume bit to zero. When the host controller receives the write to transition Force Port Resume to zero, it completes the resume sequence as defined in the USB specification, and sets both the Force Port Resume and Suspend bits to zero. Software-initiated port resumes do not affect the Port Change Detect bit in the USB_USBSTS register nor do they cause an interrupt if the Port Change Interrupt Enable bit in the USB_USBINTR register is one. An external USB event may also initiate a resume. The wake events are defined above. When a wake event occurs on a suspended port, the resume signaling is detected by the port and the resume is reflected downstream within 100 µsec. The port's Force Port Resume bit is set to one and the Port Change Detect bit in the USB_USBSTS register is set to one. If the Port Change Interrupt Enable bit in the USB_USBINTR register is one the host controller issues a hardware interrupt.

System software observes the resume event on the port, delays a port resume time (nominally 20 ms), then terminates the resume sequence by writing zero to the Force Port Resume bit in the port. The host controller receives the write of zero to Force Port Resume, terminates the resume sequence and sets Force Port Resume and Suspend port bits to zero. Software can determine that the port is enabled (not suspended) by sampling the USB_PORTSC1 register and observing that the Suspend and Force Port Resume bits are zero. Software must ensure that the host controller is running (that is HCHalted bit in the USB_USBSTS register is zero), before terminating a resume by writing zero to a

port's Force Port Resume bit. If HCHalted is one when Force Port Resume is set to zero, then SOFs do not occur down the enabled port and the device returns to suspend mode in a maximum of 10 msec.

The table below summarizes the wake-up events. Whenever a resume event is detected, the Port Change Detect bit in the USB_USBSTS register is set to one. If the Port Change Interrupt Enable bit is one in the USB_USBINTR register, the host controller generates an interrupt on the resume event. Software acknowledges the resume event interrupt by clearing the Port Change Detect status bit in the USB_USBSTS register.

Table 65-37. Behavior During Wake-up Events

Port Status and Signaling Type	Signaled Port Response	Device State	
		D0	Not D0
Port disabled, resume K-State received	No Effect	N/A	N/A
Port suspended, resume K-State received	Resume reflected downstream on signaled port. Force Port Resume status bit in USB_PORTSC1 register is set to one. Port Change Detect bit in USB_USBSTS register set to one.	[1], [2]	[2]
Port is enabled, disabled or suspended, and the port's WKDSCNNT_E bit is one. A disconnect is detected.	Depending in the initial port state, theUSB_ PORTSC1 Connected Enable status bits are set to zero, and the Connect Change status bit is set to one. Port Change Detect bit in the USB_USBSTS register is set to one.	[1], [2]	[2]
Port is enabled, disabled or suspended, and the port's WKDSCNNT_E bit is zero. A disconnect is detected.	Depending on the initial port state, the USB_PORTSC1 Connect and Enable status bits are set to zero, and the Connect Change status bit is set to one. Port Change Detect bit in the USB_USBSTS register is set to one.	[1], [3]	[3]
Port is not connected and the port's WKCNNT_E bit is one. A connect is detected.	USB_PORTSC1 Connect Status and Connect Status Change bits are set to one. Port Change Detect bit in the USB_USBSTS register is set to one.	[1], [2]	[2]
Port is not connected and the port's WKCNNT_E bit is zero. A connect is detected.	USB_PORTSC1 Connect Status and Connect Status Change bits are set to one. Port Change Detect bit in the USB_USBSTS register is set to one.	[1], [3]	[3]
Port is connected and the port's WKOC_E bit is one. An over-current condition occurs.	USB_PORTSC1 Over-current Active, Over-current Change bits are set to one. If Port Enable/Disable bit is one, it is set to zero. Port Change Detect bit in the USB_USBSTS register is set to one	[1], [2]	[2]
Port is connected and the port's WKOC_E bit is zero. An over-current condition occurs.	USB_PORTSC1 Over-current Active, Over-current Change bits are set to one. If Port Enable/Disable bit is one, it is set to zero. Port Change Detect bit in the USB_USBSTS register is set to one.	[1], [3]	[3]

[1] Hardware interrupt issued if Port Change Interrupt Enable bit in the USB_USBINTR register is one.

[2] PME# asserted if enabled (Note: PME Status must always be set to one).

[3] PME# not asserted.

65.4.3.4 Schedule Traversal Rules

The host controller executes transactions for devices using a simple, shared-memory schedule.

The schedule is comprised of a few data structures, organized into two distinct lists. The data structures are designed to provide the maximum flexibility required by USB, minimize memory traffic and hardware / software complexity.

System software maintains two schedules for the host controller: a periodic schedule and an asynchronous schedule. The root of the periodic schedule is the USB_PERIODICLISTBASE register (see Frame List Base Address (USBC_*n*_PERIODICLISTBASE))/ Device Address (USBC_*n*_DEVICEADDR). The USB_PERIODICLISTBASE register is the physical memory base address of the periodic frame list. The periodic frame list is an array of physical memory pointers. The objects referenced from the frame list must be valid schedule data structures as defined in Host Data Structures. In each micro-frame, if the periodic schedule is enabled (see Periodic scheduling threshold) then the host controller must execute from the periodic schedule before executing from the asynchronous schedule. It only executes from the asynchronous schedule after it encounters the end of the periodic schedule. The host controller traverses the periodic schedule by constructing an array offset reference from the USB_PERIODICLISTBASE and the USB_FRINDEX registers (see the following figure). It fetches the element and begins traversing the graph of linked schedule data structures.

The end of the periodic schedule is identified by a next link pointer of a schedule data structure having its T-bit set to one. When the host controller encounters a T-Bit set to one during a horizontal traversal of the periodic list, it interprets this as an End-Of-Periodic-List mark. This causes the host controller to cease working on the periodic schedule and transitions immediately to traversing the asynchronous schedule. After the transition, the host controller executes from the asynchronous schedule until the end of the micro-frame.

The following figure illustrates the derivation of pointer into frame list array.

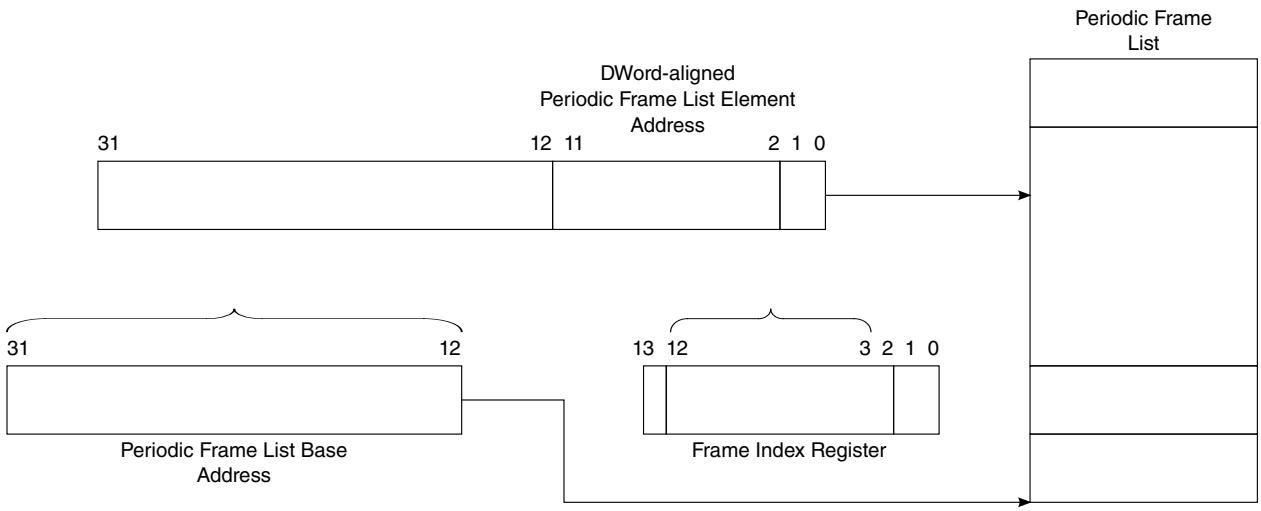


Figure 65-7. Derivation of Pointer into Frame List Array

When the host controller determines that it is the time to execute from the asynchronous list, it uses the operational register USB_ASYNCLISTADDR to access the asynchronous schedule, see the figure below.

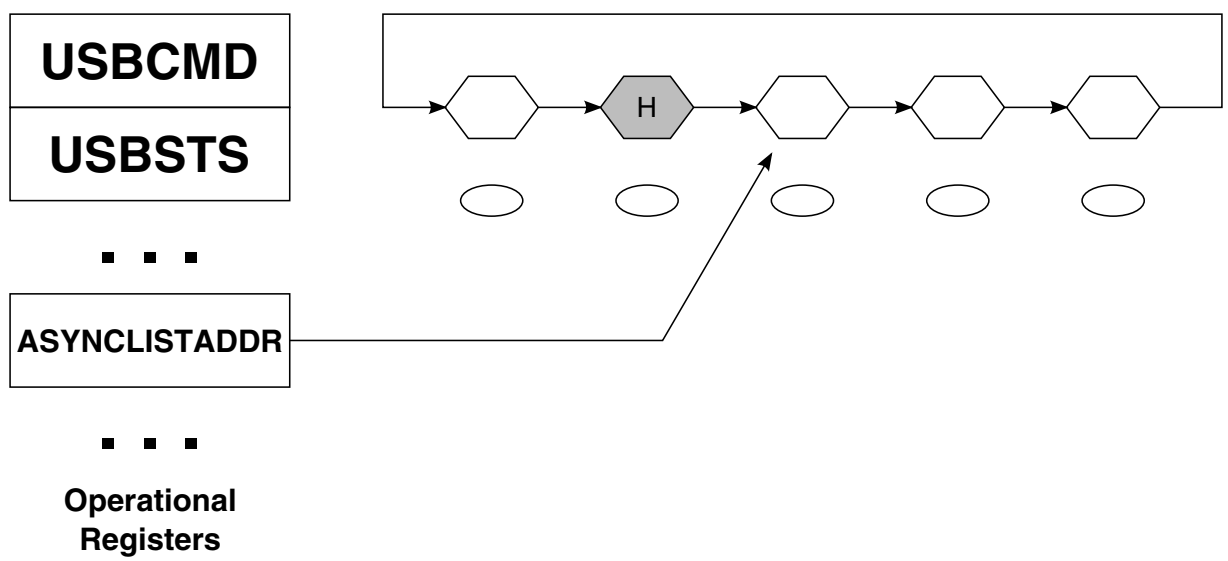


Figure 65-8. General Format of Asynchronous Schedule List

The USB_ASYNCLISTADDR register contains a physical memory pointer to the next queue head. When the host controller makes a transition to executing the asynchronous schedule, it begins by reading the queue head referenced by the USB_ASYNCLISTADDR register. Software must set queue head horizontal pointer T-bits to zero for queue heads in the asynchronous schedule. See Asynchronous Schedule for complete operational details.

65.4.3.4.1 Example - Preserving Micro-Frame Integrity

One of the requirements of a USB host controller is to maintain Frame Integrity. This means that the HC must preserve the micro-frame boundaries.

For example, SOF packets must be generated on time (within the specified allowable jitter), and High-speed EOF1,2 thresholds must be enforced. The end of micro-frame timing points EOF1 and EOF2 are clearly defined in the USB Specification Revision 2.0. One implication of this responsibility is that the HC must ensure that it does not start transactions that do not complete before the end of the micro-frame. More precisely, no transactions should be started by the host controller, which do not complete in their entirety before the EOF1 point. In order to enforce this rule, the host controller must check each transaction before it starts to ensure that it completes before the end of the micro-frame.

So, what exactly needs to be involved in this check? Fundamentally, the transaction data payload, plus bit stuffing, plus transaction overhead must be taken into consideration. It is possible to be extremely accurate on how much time the next transaction takes. Take OUTs for an example. The host controller must fetch all of the OUT data from memory in order to send it onto the USB bus. A host controller implementation could pre-fetch all of the OUT data, and pre-compute the actual number of bits in the token and data packets. In addition, the system knows the depth of the target endpoint, so it could closely estimate turnaround time for handshake. In addition, the host controller knows the size of a handshake packet. Pre-computing effects of bit stuffing and summing up the other overhead numbers can allow the host controller to know exactly whether there is enough bus time, before EOF1 to complete the OUT transaction. To accomplish this particular approach takes an inordinate amount of time and hardware complexity.

The alternative is to make a reasonable guess whether the next transaction can be started. An example approximation algorithm is described below. This example algorithm relies on the EHCI policy that periodic transactions are scheduled first in the micro-frame. It is a reasonable assumption that software never over-commits the micro-frame to periodic transactions greater than the specification allowable 80%. In the available remaining 20% bandwidth, the host controller has some ability (in this example) to decide whether or not to execute a transaction. The result of this algorithm is that sometimes, under some circumstances a transaction is not executed that could have been executed. However, under all circumstances, a transaction is never started unless there is enough time in the frame to complete the transaction.

65.4.3.4.1.1 Transaction Fit - A Best-Fit Approximation Algorithm

A curve is calculated which represents the latest start time for every packet size, at which software schedules the start of a periodic transaction.

This curve is the 80% bandwidth curve. Another curve is calculated which is the absolute, latest permitted start time for every packet size. This curve represents the absolute latest time, that a transaction of each packet size can be started and completed, in the micro-frame. A plot of these two curves are illustrated in Figure 65-9. The plot Y-axis represents the number of byte-times left in a frame.

The space between the 80% and the Last Start plots is bandwidth reclamation area. In this algorithm the host controller may skip transactions during this time if it is prudent.

The Best-Fit Approximation method plots a function ($f(x)$) between the 80% and Last Start curves. The function $f(x)$ adds a constant to every transaction's maximum packet size and the result compared with the number of bytes left in the frame. The constant represents an approximation of the effects of bit stuffing and protocol overhead. The host controller starts transactions whose results land above the function curve. The host controller will not start transactions whose results land below the function curve.

The following figure illustrates the Best-Fit Approximation.

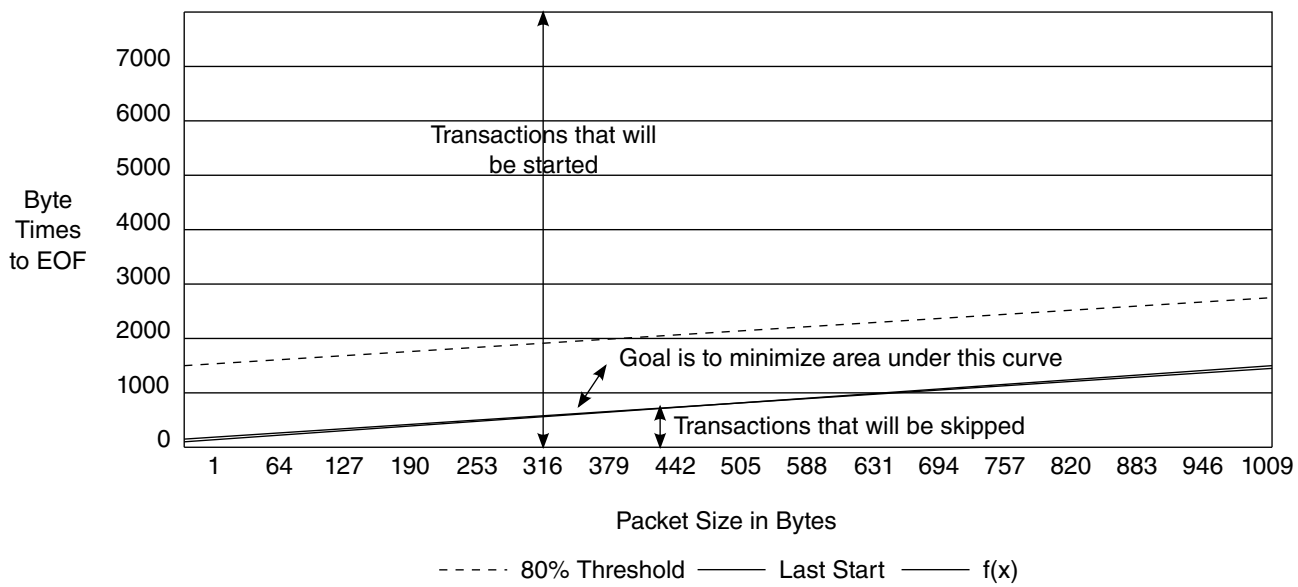


Figure 65-9. Best Fit Approximation

The LastStart line was calculated in this example to assume the absolute worst-case bus overhead per transaction. The particular transaction used is a start-split, zero-length OUT transaction with a handshake. Summaries of the component parts are listed in the table below. The component times were derived from the protocol timings defined in the USB Specification Revision 2.0.

Table 65-38. Example Worse-case Transaction Timing Components

Component	Bit time	Byte Time	Explanation

Table continues on the next page...

Table 65-38. Example Worse-case Transaction Timing Components (continued)

Split Token	76	9.5	Split token as defined in USB core specification. Includes sync, token, eop, and so on.
Host 2 Host IPG	88	11	Number of bit times required between consecutive host packets.
Token	67	8.375	Token as defined in USB core specification. Includes sync, token, eop, and so on.
Host 2 Host IPG	88	11	Token as defined in USB core specification. Includessync, token, eop, and so on.
Data Packet (0 data bytes)	66.7	8.34	Zero-length data packet. Includes sync, PID, crc16, eop, and so on.
Turnaround time	721	90.125	Time for packet initiator (Host) to see the beginning of a response to a transmitted packet.
Handshake packet	48	6	Handshake packet as defined in USB core specification. Includes sync, PID, eop, and so on.
		144	Total

The exact details of the function (f(x)) are up to the particular implementation. However, it should be obvious that the goal is to minimize the area under the curve between the approximation function and the Last Start curve, without dipping below the LastStart line, while at the same time keeping the check as simple as possible for hardware implementation. The f(x) in Figure 65-9 was constructed using the following pseudo-code test on each transaction size data point. This algorithm assumes that the host controller keeps track of the remaining bits in the frame.

```
Alorithm CheckTransactionWillFit (MaximumPacketSize, HC_BytesLeftInFrame)
Begin
Local Temp = MaximumPacketSize + 192
Local rvalue = TRUE
If MaximumPacketSize >= 128 then
     Temp += 128
End If
If Temp > HC_BytesLeftInFrame then
     Rvalue = FALSE
End If
Return rvalue
End
```

This algorithm takes two inputs, the current maximum packet size of the transaction and the hardware counter of the number of bytes left in the current micro-frame. It unconditionally adds a simple constant of 192 to the maximum packet size to account for a first-order effect of transaction overhead and bit stuffing. If the transaction size is greater than or equal to 128 bytes, then an additional constant of 128 is added to the running sum to account for the additional worst-case bit stuffing of payloads larger than 128. An inflection point was inserted at 128 because the $f(x)$ plot was getting close to the LastStart line.

65.4.3.5 Periodic Schedule Frame Boundaries vs Bus Frame Boundaries

The USB Specification Revision 2.0 requires that the frame boundaries (SOF frame number changes) of the high-speed bus and the full- and low-speed bus(s) below USB 2.0 Hubs be strictly aligned.

Super-imposed on this requirement is that USB 2.0 Hubs manage full- and low-speed transactions through a micro-frame pipeline (see start- (SS) and complete- (CS) splits illustrated in the following figure). A simple, direct projection of the frame boundary model into the host controller interface schedule architecture creates tension (complexity for both hardware and software) between the frame boundaries and the scheduling mechanisms required to service the full- and low-speed transaction translator periodic pipelines.

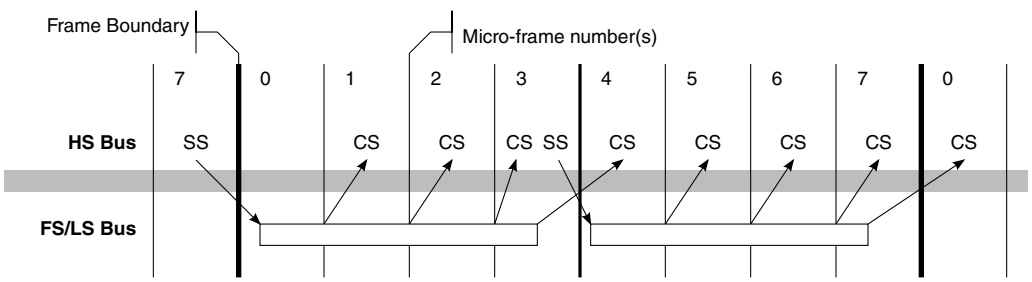


Figure 65-10. Frame Boundary Relationship between HS bus and FS/LS Bus

The simple projection, as the above figure illustrates, introduces frame-boundary wrap conditions for scheduling on both the beginning and end of a frame. In order to reduce the complexity for hardware and software, the host controller is required to implement one micro-frame phase shift for its view of frame boundaries. The phase shift eliminates the beginning of frame and frame-wrap scheduling boundary conditions.

The implementation of this phase shift requires that the host controller use one register value for accessing the periodic frame list and another value for the frame number value included in the SOF token. These two values are separate, but tightly coupled. The periodic frame list is accessed through the Frame List Index Register (USB_FRINDEX) documented in USB Frame Index (USBC_*n*_FRINDEX) and initially illustrated in Schedule Traversal Rules . Bits FRINDEX[2:0], represent the micro-frame number. The SOF value is coupled to the value of FRINDEX[13:3]. Both FRINDEX[13:3] and the SOF value are increment based on FRINDEX[2:0]. It is required that the SOF value be delayed from the FRINDEX value by one micro-frame. The one micro-frame delay yields host controller periodic schedule and bus frame boundary relationship as illustrated in the following figure. This adjustment allows software to trivially schedule the periodic

start and complete-split transactions for full-and low-speed periodic endpoints, using the natural alignment of the periodic schedule interface. The reasons for selecting this phase-shift are beyond the scope of this specification.

The following figure illustrates how periodic schedule data structures relate to schedule frame boundaries and bus frame boundaries. To aid the presentation, two terms are defined: The host controller's view of the 1 msec boundaries is called H-Frames. The high-speed bus's view of the 1 msec boundaries is called B-Frames.

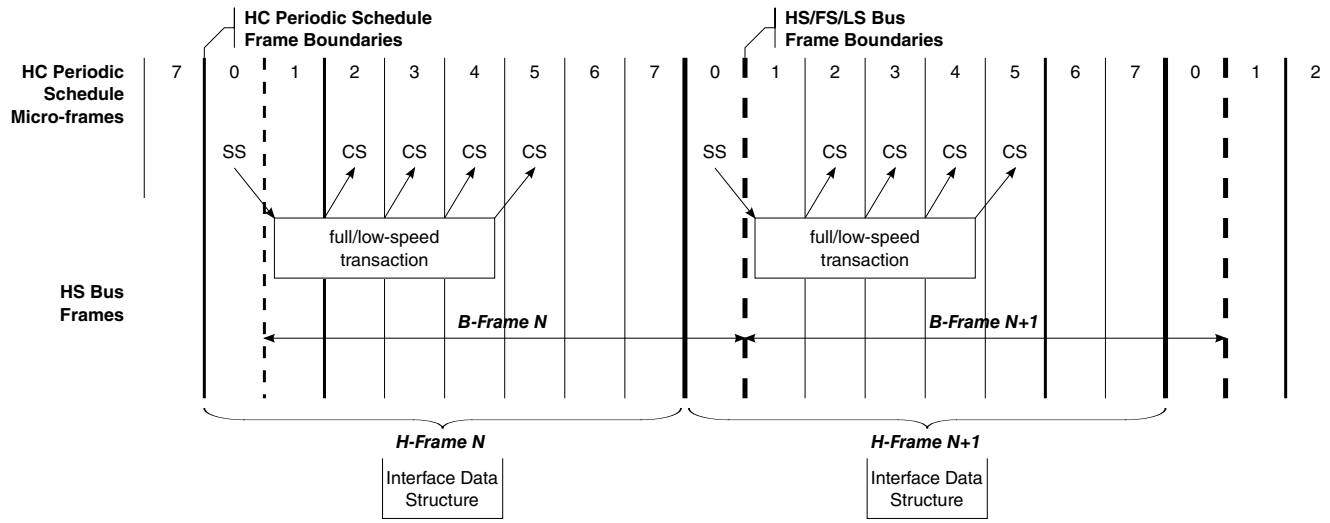


Figure 65-11. Relationship of Periodic Schedule Frame Boundaries to Bus Frame Boundaries

H-Frame boundaries for the host controller correspond to increments of FRINDEX[13:3]. Micro-frame numbers for the H-Frame are tracked by FRINDEX[2:0]. B-Frame boundaries are visible on the high-speed bus through changes in the SOF token's frame number. Micro-frame numbers on the high-speed bus are only derived from the SOF token's frame number (that is the high-speed bus sees eight SOFs with the same frame number value). H-Frames and B-Frames have the fixed relationship (that is B-Frames lag H-Frames by one micro-frame time) illustrated in the figure above. The host controller's periodic schedule is naturally aligned to H-Frames. Software schedules transactions for full- and low-speed periodic endpoints relative the H-Frames. The result is these transactions execute on the high-speed bus at exactly the right time for the USB 2.0 Hub periodic pipeline. As described in USB Frame Index (USBC_*n*_FRINDEX), the SOF Value can be implemented as a shadow register (in this example, called SOFV), which lags the FRINDEX register bits [13:3] by one micro-frame count. This lag behavior can be accomplished by incrementing FRINDEX[13:3] based on carry-out on the 7 to 0 increment of FRINDEX[2:0] and incrementing SOFV based on the transition of 0 to 1 of FRINDEX[2:0].

Software is allowed to write to FRINDEX. USB Frame Index (USBC_*n*_FRINDEX) provides the requirements that software should adhere when writing a new value in FRINDEX.

The table below illustrates the required relationship between the value of FRINDEX and the value of SOFV.

Table 65-39. Operation of FRINDEX and SOFV (SOF Value Register)

Current			Next		
FRINDEX[F]	SOFV	FRINDEX[mF]	FRINDEX[F]	SOFV	FRINDEX[mF]
N	N	111b	N+1	N	000b
N+1	N	000b	N+1	N+1	001b
N+1	N+1	001b	N+1	N+1	010b
N+1	N+1	010b	N+1	N+1	011b
N+1	N+1	011b	N+1	N+1	100b
N+1	N+1	100b	N+1	N+1	101b
N+1	N+1	101b	N+1	N+1	110b
N+1	N+1	110b	N+1	N+1	111b

NOTE

Where [F] = [13:3]; [µF] = [2:0]

65.4.3.6 Periodic Schedule

The periodic schedule traversal is enabled or disabled through the Periodic Schedule Enable bit in the USB_USBCMD register.

If the Periodic Schedule Enable bit is set to zero, then the host controller simply does not try to access the periodic frame list through the USB_PERIODICLISTBASE register. Likewise, when the Periodic Schedule Enable bit is one, then the host controller does use the USB_PERIODICLISTBASE register to traverse the periodic schedule. The host controller will not react to modifications to the Periodic Schedule Enable immediately. In order to eliminate conflicts with split transactions, the host controller evaluates the Periodic Schedule Enable bit only when FRINDEX[2:0] is zero. System software must not disable the periodic schedule if the schedule contains an active split transaction work item that spans the 000b micro-frame. These work items must be removed from the schedule before the Periodic Schedule Enable bit is written to zero. The Periodic Schedule Status bit in the USB_USBSTS register indicates status of the periodic schedule. System software enables (or disables) the periodic schedule by writing one (or zero) to the Periodic Schedule Enable bit in the USB_USBCMD register. Software then can poll the Periodic Schedule Status bit to determine when the periodic schedule has

made the desired transition. Software must not modify the Periodic Schedule Enable bit unless the value of the Periodic Schedule Enable bit equals that of the Periodic Schedule Status bit.

The periodic schedule is used to manage all isochronous and interrupt transfer streams. The base of the periodic schedule is the periodic frame list. Software links schedule data structures to the periodic frame list to produce a graph of scheduled data structures. The graph represents an appropriate sequence of transactions on the

The following figure illustrates isochronous transfers (using iTDs and siTDs) with a period of one are linked directly to the periodic frame list. Interrupt transfers (are managed with queue heads) and isochronous streams with periods other than one are linked following the period-one iTD/siTDs. Interrupt queue heads are linked into the frame list ordered by poll rate. Longer poll rates are linked first (for example, closest to the periodic frame list), followed by shorter poll rates, with queue heads with a poll rate of one, on the very end.

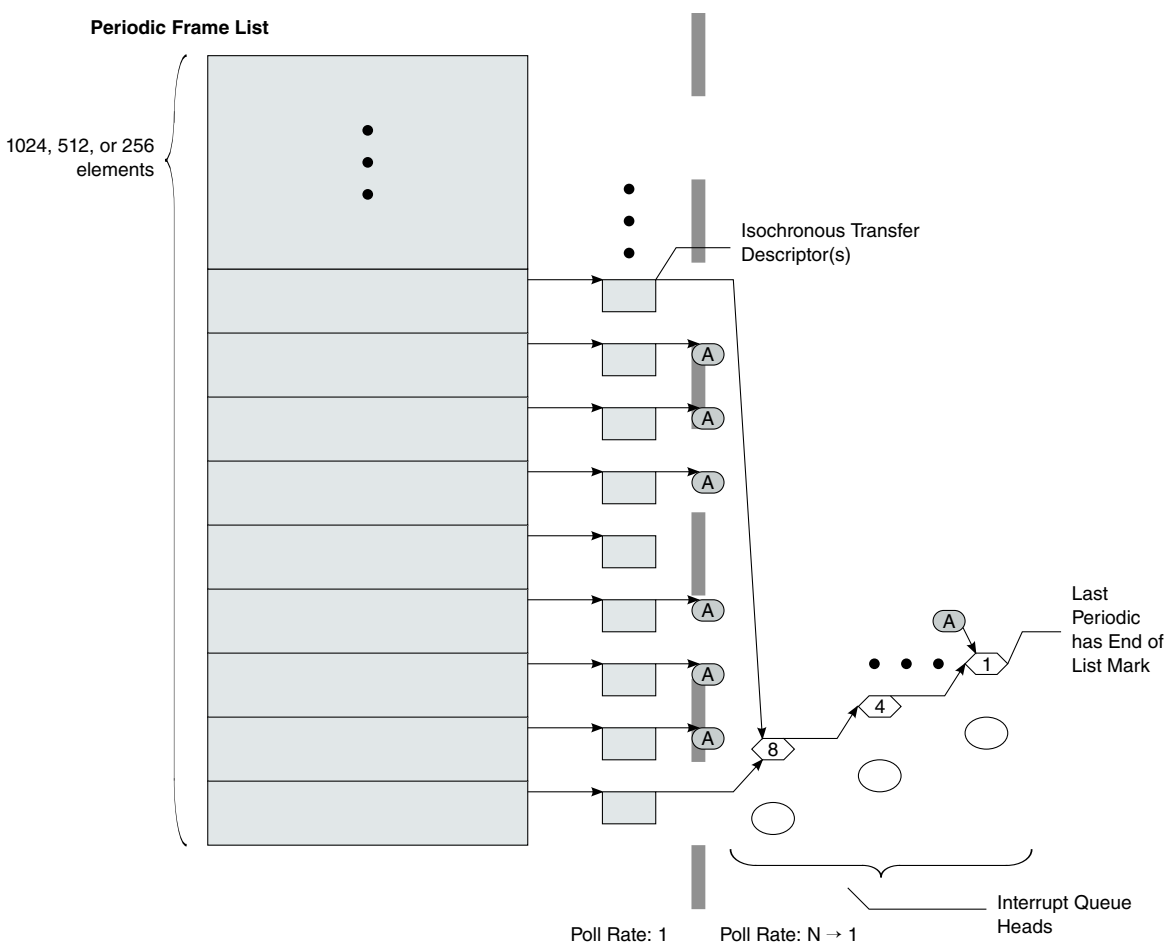


Figure 65-12. Example Periodic Schedule

65.4.3.7 Managing Isochronous Transfers Using iTDs

The structure of an iTD is presented in Isochronous (High-Speed) Transfer Descriptor (iTD). The four distinct sections to an iTD:

- The first field is the Next Link Pointer. This field is for schedule linkage purposes only.
- Transaction description array. This area is an eight-element array. Each element represents control and status information for one micro-frame's worth of transactions for a single high-speed isochronous endpoint.
- The buffer page pointer array is a 7-element array of physical memory pointers to data buffers. These are 4 K aligned pointers to physical memory.
- Endpoint capabilities. This area utilizes the unused low-order 12 bits of the buffer page pointer array. The fields in this area are used across all transactions executed for this iTD, including endpoint addressing, transfer direction, maximum packet size and high-bandwidth multiplier.

65.4.3.7.1 Host Controller Operational Model for iTDs

The host controller uses FRINDEX register bits [12:3] to index into the periodic frame list. This means that the host controller visits each frame list element eight consecutive times before incrementing to the next periodic frame list element. Each iTD contains eight transaction descriptions, which map directly to FRINDEX register bits [2:0]. Therefore, each transaction descriptor corresponds to one micro-frame. Each iTD can span 8 micro-frames worth of transactions.

When the host controller fetches an iTD, it uses FRINDEX register bits [2:0] to index into the transaction description array.

If the active bit in the Status field of the indexed transaction description is set to zero, the host controller ignores the iTD and follows the Next pointer to the next schedule data structure.

When the indexed active bit is one, the host controller continues to parse the iTD. It stores the indexed transaction description and the general endpoint information (device address, endpoint number, maximum packet size, and so on.). It also uses the Page Select (PG) field to index the buffer pointer array, storing the selected buffer pointer and the next sequential buffer pointer. For example, if PG field is 0, then the host controller stores Page 0 and Page 1.

The host controller constructs a physical data buffer address by concatenating the current buffer pointer (as selected using the current transaction description's PG field) and the transaction description's Transaction Offset field. The host controller uses the endpoint addressing information and I/O-bit to execute a transaction to the appropriate endpoint. When the transaction is complete, the host controller clears the active bit and writes back any additional status information to the Status field in the currently selected transaction description.

The data buffer associated with the iTD must be virtually contiguous memory. Seven page pointers are provided to support eight high-bandwidth transactions regardless of the starting packet's offset alignment into the first page. A starting buffer pointer (physical memory address) is constructed by concatenating the page pointer (for example, page 0 pointer) selected by the active transaction descriptions' PG (for example, value: 00B) field with the transaction offset field. As the transaction moves data, the host controller must detect when an increment of the current buffer pointer crosses a page boundary. When this occurs the host controller simply replaces the current buffer pointer's page portion with the next page pointer (for example, page 1 pointer) and continues to move data. The size of each bus transaction is determined by the value in the Maximum Packet Size field. An iTD supports high-bandwidth pipes through the Mult (multiplier) field. When the Mult field is 1, 2, or 3, the host controller executes the specified number of Maximum Packet sized bus transactions for the endpoint in the current micro-frame. In other words, the Mult field represents a transaction count for the endpoint in the current micro-frame. If the Mult field is zero, the operation of the host controller is undefined. The transfer description is used to service all transactions indicated by the Mult field.

For OUT transfers, the value of the Transaction X Length field represents the total bytes to be sent during the micro-frame. The Mult field must be set by software to be consistent with Transaction X Length and Maximum Packet Sixe. The host controller sends the bytes in Maximum Packet Size'd portions. After each transaction, the host controller decrements its local copy of Transaction X Length by Maximum Packet Size. The number of bytes the host controller sends is always Maximum Packet Size or Transaction X Length, whichever is less. The host controller advances the transfer state in the transfer description, updates the appropriate record in the iTD and moves to the next schedule data structure. The maximum sized transaction supported is 3 x 1024 bytes.

For IN transfers, the host controller issues Mult transactions. It is assumed that software has properly initialized the iTD to accommodate all of the possible data. During each IN transaction, the host controller must use Maximum Packet Size to detect packet babble errors. The host controller keeps the sum of bytes received in the Transaction X Length field. After all transactions for the endpoint have completed for the micro-frame, Transaction X Length contains the total bytes received. If the final value of Transaction X Length is less than the value of Maximum Packet Size, then less data than was allowed for was received from the associated endpoint. This short packet condition does not set

the USBINT bit in the USB_USBSTS register to one. The host controller will not detect this condition. If the device sends more than Transaction X Length or Maximum Packet Size bytes (whichever is less), then the host controller sets the Babble Detected bit to one and set the Active bit to zero. Note, that the host controller is not required to update the iTD field Transaction X Length in this error scenario. If the Mult field is greater than one, then the host controller automatically executes the value of Mult transactions. The host controller will not execute all Mult transactions if:

- The endpoint is an OUT and Transaction X Length goes to zero before all the Mult transactions have executed (ran out of data), or
- The endpoint is an IN and the endpoint delivers a short packet, or an error occurs on a transaction before Mult transactions have been executed. The end of micro-frame may occur before all of the transaction opportunities have been executed. When this happens, the transfer state of the transfer description is advanced to reflect the progress that was made, the result written back to the iTD and the host controller proceeds to processing the next micro-frame. Refer to Appendix D for a table summary of the host controller required behavior for all the high-bandwidth transaction cases.

65.4.3.7.2 Software Operational Model for iTDs

A client buffer request to an isochronous endpoint may span 1 to N micro-frames. When N is larger than one, system software may have to use multiple iTDs to read or write data with the buffer (if N is larger than eight, it must use more than one iTD).

The following figure illustrates the simple model of how a client buffer is mapped by system software to the periodic schedule (that is the periodic frame list and a set of iTDs). On the right is the client description of its request. The description includes a buffer base address plus additional annotations to identify which portions of the buffer should be used with each bus transaction. In the middle is the iTD data structures used by the system software to service the client request. Each iTD can be initialized to service up to 24 transactions, organized into eight groups of up to three transactions each. Each group maps to one micro-frame's worth of transactions. The EHCI controller does not provide per-transaction results within a micro-frame. It treats the per-micro-frame transactions as a single logical transfer. On the left is the host controller's frame list. System software establishes references from the appropriate locations in the frame list to each of the appropriate iTDs. If the buffer is large, then system software can use a small set of iTDs to service the entire buffer. System software can activate the transaction description records (contained in each iTD) in any pattern required for the particular data stream.

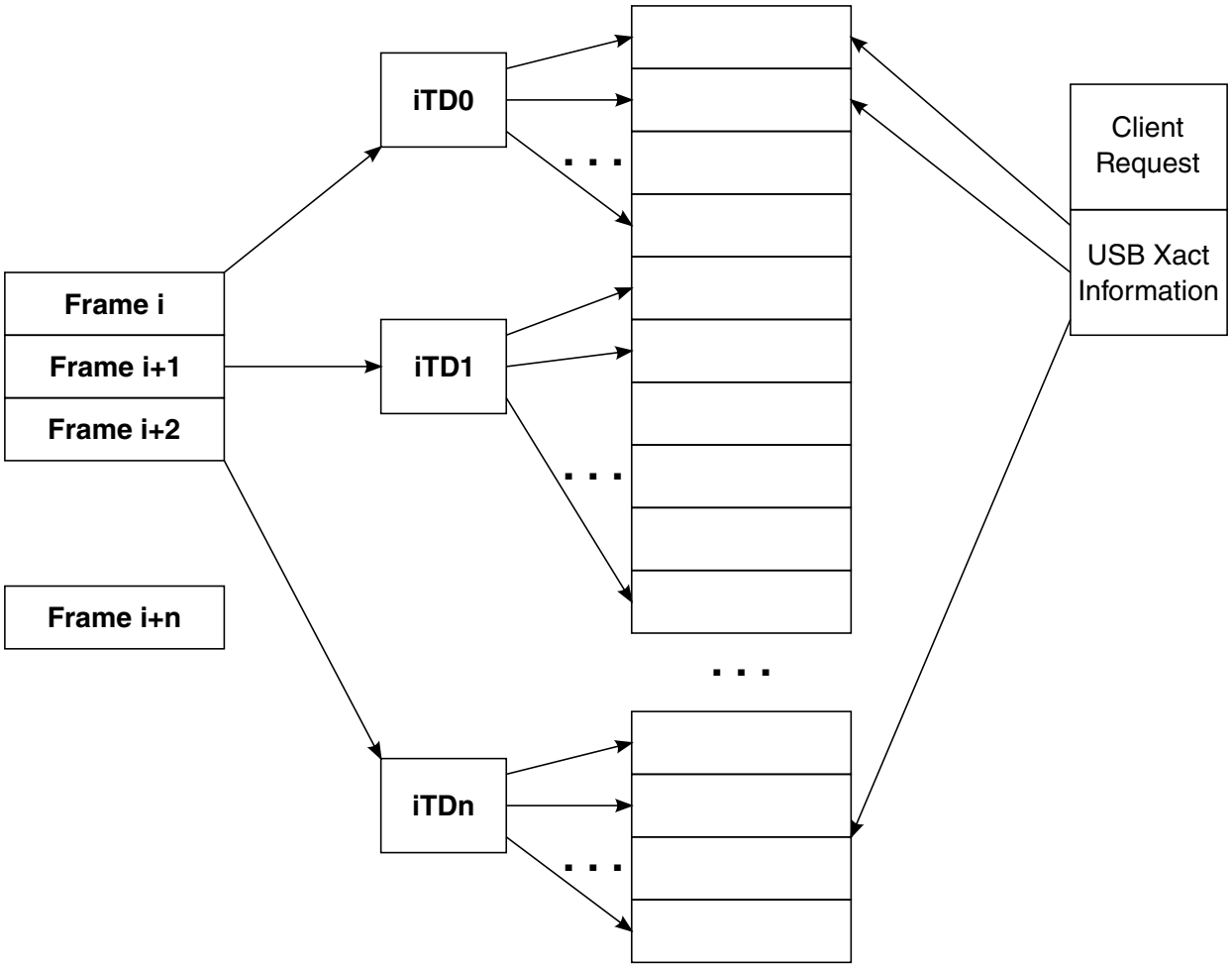


Figure 65-13. Example Association of iTDs to Client Request Buffer

As noted above, the client request includes a pointer to the base of the buffer and offsets into the buffer to annotate which buffer sections are to be used on each bus transaction that occurs on this endpoint. System software must initialize each transaction description in an iTD to ensure it uses the correct portion of the client buffer. For example, for each transaction description, the PG field is set to index the correct physical buffer page pointer and the Transaction Offset field is set relative to the correct buffer pointer page (for example, the same one referenced by the PG field). When the host controller executes a transaction it selects a transaction description record based on FRINDEX[2:0]. It then uses the current Page Buffer Pointer (as selected by the PG field) and concatenates to the transaction offset field. The result is a starting buffer address for the transaction. As the host controller moves data for the transaction, it must watch for a page wrap condition and properly advance to the next available Page Buffer Pointer. System software must not use the Page 6 buffer pointer in a transaction description where the length of the transfer wraps a page boundary. Doing so yields undefined behavior. The host controller hardware is not required to 'alias' the page selector to Page zero. USB 2.0 isochronous

endpoints can specify a period greater than one. Software can achieve the appropriate scheduling by linking iTDs into the appropriate frames (relative to the frame list) and by setting appropriate transaction description elements active bits to one.

65.4.3.7.2.1 Periodic scheduling threshold

The Isochronous Scheduling Threshold field in the USB_HCCPARAMS capability register is an indicator to system software as to how the host controller pre-fetches and effectively caches schedule data structures.

It is used by system software when adding isochronous work items to the periodic schedule. The value of this field indicates to system software the minimum distance it can update isochronous data (relative to the current location of the host controller execution in the periodic list) and still have the host controller process them.

The iTD and siTD data structures each describe 8 micro-frames worth of transactions. The host controller is allowed to cache one (or more) of these data structures in order to reduce memory traffic. Three basic caching models that account for the fact the isochronous data structures span 8 micro-frames. The three caching models are: no caching, micro-frame caching and frame caching.

When software is adding new isochronous transactions to the schedule, it always performs a read of the USB_FRINDEX register to determine the current frame and micro-frame the host controller is currently executing. Of course, there is no information about where in the micro-frame the host controller is, so a constant uncertainty-factor of one micro-frame has to be assumed. Combining the knowledge of where the host controller is executing with the knowledge of the caching model allows the definition of simple algorithms for how closely software can reliably work to the executing host controller.

No caching is indicated with a value of zero in the Isochronous Scheduling Threshold field. The host controller may pre-fetch data structures during a periodic schedule traversal (per micro-frame) but always dumps any accumulated schedule state at the end of the micro-frame. At the appropriate time relative to the beginning of every micro-frame, the host controller always begins schedule traversal from the frame list. Software can use the value of the USB_FRINDEX register (plus the constant 1 uncertainty-factor) to determine the approximate position of the executing host controller. When no caching is selected, software can add an isochronous transaction as near as 2 micro-frames in front of the current executing position of the host controller.

Frame caching is indicated with a non-zero value in bit [7] of the Isochronous Scheduling Threshold field. In the frame-caching model, system software assumes that the host controller caches one (or more) isochronous data structures for an entire frame (8 micro-frames). Software uses the value of the USB_FRINDEX register (plus the constant 1

uncertainty) to determine the current micro-frame/frame (assume modulo 8 arithmetic in adding the constant 1 to the micro-frame number). For any current frame N, if the current micro-frame is 0 to 6, then software can safely add isochronous transactions to Frame N + 1. If the current micro-frame is 7, then software can add isochronous transactions to Frame N + 2.

Micro-frame caching is indicated with a non-zero value in the least-significant 3 bits of the Isochronous Scheduling Threshold field. System software assumes the host controller caches one or more periodic data structures for the number of micro-frames indicated in the Isochronous Scheduling Threshold field. For example, if the count value were 2, then the host controller keeps a window of 2 micro-frames worth of state (current micro-frame, plus the next) on-chip. On each micro-frame boundary, the host controller releases the current micro-frame state and begins accumulating the next micro-frame state.

65.4.3.8 Asynchronous Schedule

The Asynchronous schedule traversal is enabled or disabled through the Asynchronous Schedule Enable bit in the USB_USBCMD register.

If the Asynchronous Schedule Enable bit is set to zero, then the host controller simply does not try to access the asynchronous schedule through the USB_ASYNCLISTADDR register. Likewise, when the Asynchronous Schedule Enable bit is one, then the host controller does use the USB_ASYNCLISTADDR register to traverse the asynchronous schedule. Modifications to the Asynchronous Schedule Enable bit are not necessarily immediate. Rather the new value of the bit is taken into consideration the next time the host controller needs to use the value of the USB_ASYNCLISTADDR register to get the next queue head.

The Asynchronous Schedule Status bit in the USB_USBSTS register indicates status of the asynchronous schedule. System software enables (or disables) the asynchronous schedule by writing one (or zero) to the Asynchronous Schedule Enable bit in the USB_USBCMD register. Software then can poll the Asynchronous Schedule Status bit to determine when the asynchronous schedule has made the desired transition. Software must not modify the Asynchronous Schedule Enable bit unless the value of the Asynchronous Schedule Enable bit equals that of the Asynchronous Schedule Status bit.

The asynchronous schedule is used to manage all Control and Bulk transfers. Control and Bulk transfers are managed using queue head data structures. The asynchronous schedule is based at the USB_ASYNCLISTADDR register. The default value of the USB_ASYNCLISTADDR register after reset is undefined and the schedule is disabled when the Asynchronous Schedule Enable bit is zero.

Software may only write this register with defined results when the schedule is disabled. For example, Asynchronous Schedule Enable bit in the USB_USBCMD and the Asynchronous Schedule Status bit in the USB_USBSTS register are zero. System software enables execution from the asynchronous schedule by writing a valid memory address (of a queue head) into this register. Then software enables the asynchronous schedule by setting the Asynchronous Schedule Enable bit to one. The asynchronous schedule is actually enabled when the Asynchronous Schedule Status bit is one.

When the host controller begins servicing the asynchronous schedule, it begins by using the value of the USB_ASYNCLISTADDR register. It reads the first referenced data structure and begins executing transactions and traversing the linked list as appropriate. When the host controller completes processing the asynchronous schedule, it retains the value of the last accessed queue head's horizontal pointer in the USB_ASYNCLISTADDR register. Next time the asynchronous schedule is accessed, this is the first data structure that is serviced. This provides round-robin fairness for processing the asynchronous schedule.

A host controller completes processing the asynchronous schedule when one of the following events occur:

- The end of a micro-frame occurs.
- The host controller detects an empty list condition (see Empty Asynchronous Schedule Detection)
- The schedule has been disabled through the Asynchronous Schedule Enable bit in the USB_USBCMD register.

The queue heads in the asynchronous list are linked into a simple circular list as shown in Figure 65-8. Queue head data structures are the only valid data structures that may be linked into the asynchronous schedule. An isochronous transfer descriptor (iTD or siTD) in the asynchronous schedule yields undefined results.

The maximum packet size field in a queue head is sized to accommodate the use of this data structure for all non-isochronous transfer types. The USB Specification, Revision 2.0 specifies the maximum packet sizes for all transfer types and transfer speeds. System software should always parameterize the queue head data structures according to the core specification requirements.

65.4.3.8.1 Adding Queue Heads to Asynchronous Schedule

This is a software requirement section.

There are two independent events for adding queue heads to the asynchronous schedule. The first is the initial activation of the asynchronous list. The second is inserting a new queue head into an activated asynchronous list.

Activation of the list is simple. System software writes the physical memory address of a queue head into the USB_ASYNCLISTADDR register, then enables the list by setting the Asynchronous Schedule Enable bit in the USB_USBCMD register to one.

When inserting a queue head into an active list, software must ensure that the schedule is always coherent from the host controllers' point of view. This means that the system software must ensure that all queue head pointer fields are valid. For example, qTD pointers have T-Bits set to one or reference valid qTDs and the Horizontal Pointer references a valid queue head data structure. The following algorithm represents the functional requirements:

```
InsertQueueHead (pQHeadCurrent, pQueueHeadNew)
          --
          -- Requirement: all inputs must be properly initialized.
          --
          -- pQHeadCurrent is a pointer to a queue head that is
          -- already in the active list
          -- pQHeadNew is a pointer to the queue head to be added
          --
          -- This algorithm links a new queue head into a existing
          -- list
          --
          pQueueHeadNew.HorizontalPointer  =  pQueueHeadCurrent.HorizontalPointer
          pQueueHeadCurrent.HorizontalPointer =  physicalAddressOf(pQueueHeadNew)
End InsertQueueHead
```

65.4.3.8.2 Removing Queue Heads from Asynchronous Schedule

This is a software requirement section.

There are two independent events for removing queue heads from the asynchronous schedule. The first is shutting down (deactivating) the asynchronous list. The second is extracting a single queue head from an activated list.

Software deactivates the asynchronous schedule by setting the Asynchronous Schedule Enable bit in the USB_USBCMD register to zero. Software can determine when the list is idle when the Asynchronous Schedule Status bit in the USB_USBSTS register is zero. The normal mode of operation is that software removes queue heads from the asynchronous schedule without shutting it down. Software must not remove an active queue head from the schedule. Software should first deactivate all active qTDs, wait for the queue head to go inactive, then remove the queue head from the asynchronous list. Software removes a queue head from the asynchronous list through the following algorithm. As illustrated, the unlinking is quite easy. Software merely must ensure all of the link pointers reachable by the host controller are kept consistent.

```
UnlinkQueueHead (pQHeadPrevious, pQueueHeadToUnlink, pQHeadNext)
          --
          -- Requirement: all inputs must be properly initialized.
          --
          -- pQHeadPrevious is a pointer to a queue head that
          -- references the queue head to remove
          -- pQHeadToUnlink is a pointer to the queue head to be
```

```
          -- removed
          -- pQheadNext is a pointer to a queue head still in the
          -- schedule. Software provides this pointer with the
          -- following strict rules:
          --          if the host software is one queue head, then
          --          pQHeadNext must be the same as
          --          QueueheadToUnlink.HorizontalPointer. If the host
          --          software is unlinking a consecutive series of
          --          queue heads, QHeadNext must be set by software to
          --          the  queue head remaining in the schedule.
          --
          -- This algorithm unlinks a queue head from a circular list
          --
          pQueueHeadPrevious.HorizontalPointer =  pQueueHeadToUnlink.HorizontalPointer
                                  pQueueHeadToUnlink.HorizontalPointer = pQHeadNext
End UnlinkQueueHead
```

If software removes the queue head with the H-bit set to one, it must select another queue head still linked into the schedule and set its H-bit to one. This should be completed before removing the queue head. The requirement is that software keep one queue head in the asynchronous schedule, with its H-bit set to one. At the point software has removed one or more queue heads from the asynchronous schedule, it is unknown whether the host controller has a cached pointer to them. Similarly, it is unknown how long the host controller might retain the cached information, as it is implementation dependent and may be affected by the actual dynamics of the schedule load. Therefore, once software has removed a queue head from the asynchronous list, it must retain the coherency of the queue head (link pointers, and so on). It cannot disturb the removed queue heads until it knows that the host controller does not have a local copy of a pointer to any of the removed data structures.

The method software uses to determine when it is safe to modify a removed queue head is to handshake with the host controller. The handshake mechanism allows software to remove items from the asynchronous schedule, then execute a simple, lightweight handshake that is used by software as a key that it can free (or reuse) the memory associated the data structures it has removed from the asynchronous schedule.

The handshake is implemented with three bits in the host controller. The first bit is a command bit (Interrupt on Async Advance Doorbell bit in the USB_USBCMD register) that allows software to inform the host controller that something has been removed from its asynchronous schedule. The second bit is a status bit (Interrupt on Async Advance bit in the USB_USBSTS register) that the host controller sets after it has released all on-chip state that may potentially reference one of the data structures just removed. When the host controller sets this status bit to one, it also sets the command bit to zero. The third bit is an interrupt enable (Interrupt on Async Advance bit in the USB_USBINTR register) that is matched with the status bit. If the status bit is one and the interrupt enable bit is one, then the host controller asserts a hardware interrupt.

The figure below illustrates a general example. In this example, consecutive queue heads (B and C) are unlinked from the schedule using the algorithm above. Before the unlink operation, the host controller has a copy of queue head A.

The unlink algorithm requires that as software unlinks each queue head, the unlinked queue head is loaded with the address of a queue head that remains in the asynchronous schedule.

When the host controller observes that doorbell bit being set to one, it makes a note of the local reachable schedule information. In this example, the local reachable schedule information includes both queue heads (A and B). It is sufficient that the host controller can set the status bit (and clear the doorbell bit) as soon as it has traversed beyond current reachable schedule information (that is traversed beyond queue head (B) in this example). The following figure illustrates the generic queue head unlink scenario.

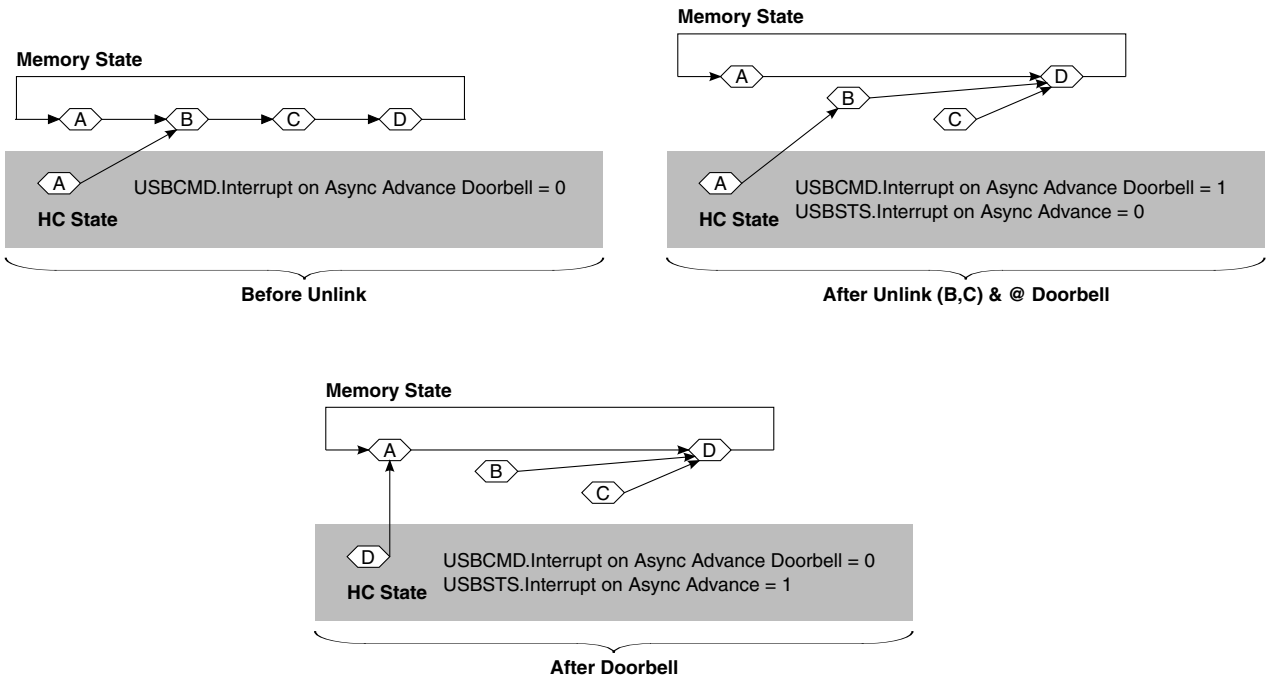


Figure 65-14. Generic Queue Head Unlink Scenario

Alternatively, a host controller implementation is allowed to traverse the entire asynchronous schedule list (for example, observed the head of the queue (twice)) before setting the Advance on Async status bit to one.

Software may re-use the memory associated with the removed queue heads after it observes the Interrupt on Async Advance status bit is set to one, following assertion of the doorbell. Software should acknowledge the Interrupt on Async Advance status as indicated in the USB_USBSTS register, before using the doorbell handshake again.

65.4.3.8.3 Empty Asynchronous Schedule Detection

The Enhanced Host Controller Interface uses two bits to detect when the asynchronous schedule is empty.

The queue head data structure (see Table 65-25) defines an *H-bit* in the queue head, which allows software to mark a queue head as being the *head* of the reclaim list. The Enhanced Host Controller Interface also keeps a 1-bit flag in the USB_USBSTS register (*Reclamation*) that is set to zero when the Enhanced Interface Host Controller observes a queue head with the H-bit set to one. The reclamation flag in the status register is set to one when any USB transaction from the asynchronous schedule is executed (or whenever the asynchronous schedule starts, see Asynchronous schedule traversal: *Start Event*).

If the Enhanced Host Controller Interface ever encounters an *H-bit* of one and a *Reclamation* bit of zero, the EHCI controller simply stops traversal of the asynchronous schedule.

An example illustrating the H-bit in a schedule is shown in the following figure.

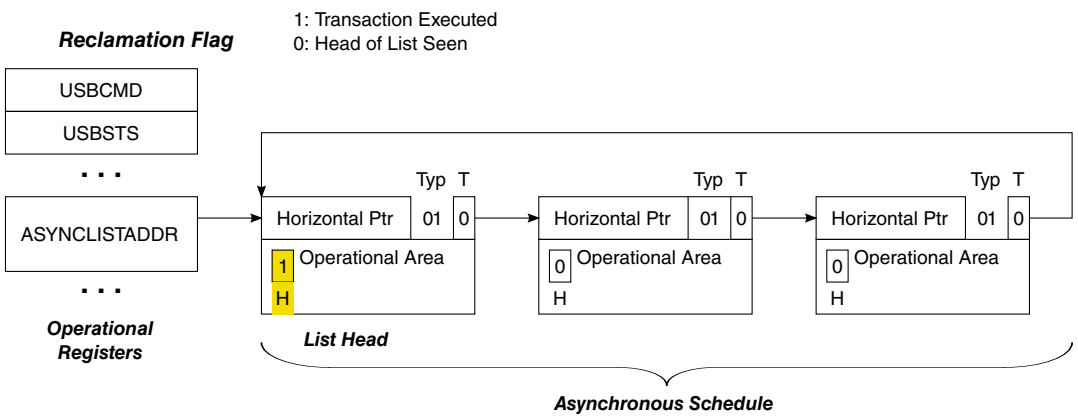


Figure 65-15. Asynchronous Schedule List w/Annotation to Mark Head of List

Software must ensure there is at most one queue head with the *H-bit* set to one, and that it is always coherent with respect to the schedule.

65.4.3.8.4 Restarting Asynchronous Schedule Before EOF

There are many situations where the host controller will detect an empty list *long* before the end of the micro-frame.

It is important to remember that under many circumstances the schedule traversal has stopped due to Nak/Nyet responses from all endpoints.

An example of particular interest is when a start-split for a bulk endpoint occurs early in the micro-frame. Given the EHCI simple traversal rules, the complete-split for that transaction may Nak/Nyet out very quickly. If it is the only item in the schedule, then the host controller ceases traversal of the Asynchronous schedule very early in the micro-frame. In order to provide reasonable service to this endpoint, the host controller should issue the complete-split before the end of the current micro-frame, instead of waiting

until the next micro-frame. When the reason for host controller idling asynchronous schedule traversal is because of empty list detection, it is mandatory the host controller implement a 'waking' method to resume traversal of the asynchronous schedule. An example method is described below.

65.4.3.8.4.1 Example Method for Restarting Asynchronous Schedule Traversal

The reason for idling the host controller when the list is empty is to keep the host controller from unnecessarily occupying too much memory bandwidth. The question is: *how long should the host controller stay idle before restarting*?

The answer in this example is based on deriving a manifest constant, which is the amount of time the host controller will stay idle before restarting traversal. In this example, the manifest constant is called *AsyncSchedSleepTime*, and has a value of 10 µsec. The value is derived based on the analysis in Example Derivation for *AsyncSchedSleepTime*, The traversal algorithm is simple:

- Traverse the Asynchronous schedule until the either an End-Of-micro-Frame event occurs, or an empty list is detected. If the event is an End-of-micro-Frame, go attempt to traverse the Periodic schedule. If the event is an empty list, then set a sleep timer and go to a *schedule sleep* state.
- When the sleep timer expires, set working context to the Asynchronous Schedule start condition and go to *schedule active* state. The start context allows the HC to reload *Nakcnt* fields, and so on. So the HC has a chance to run for more than one iteration through the schedule.

This process simply repeats itself each micro-frame. The figure below illustrates a sample state machine to manage the active and sleep states of the Asynchronous Schedule traversal policy. There are three states: Actively traversing the Asynchronous schedule, Sleeping, and Not Active. The last two are similar in terms of interaction with the Asynchronous schedule, but the Not Active state means that the host controller is busy with the Periodic schedule or the Asynchronous schedule is not enabled. The Sleeping state is specifically a special state where the host controller is just waiting for a period of time before resuming execution of the Asynchronous schedule.

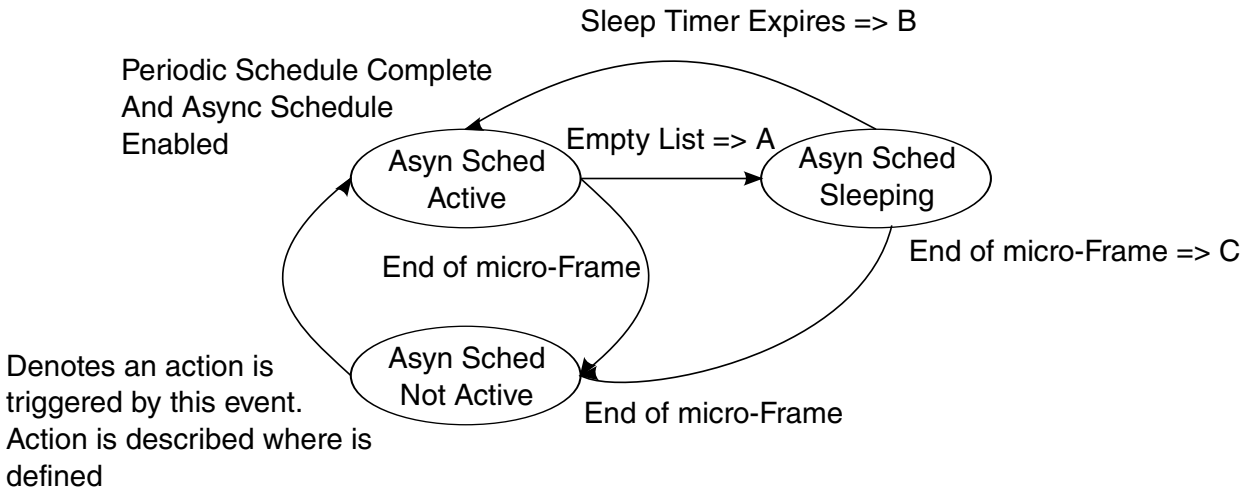


Figure 65-16. Example State Machine for Managing Asynchronous Schedule Traversal

The actions referred to in the figure above are defined in the following table.

Table 65-40. Asynchronous Schedule SM Transition Actions

Action	Action Description Label
A	On detection of the empty list, the host controller sets the *AsynchronousTraversalSleepTimer* to *AsyncSchedSleepTime*.
B	When the *AsynchronousTraversalSleepTimer* expires, the host controller sets the *Reclamation* bit in the USBSTS register to one and moves the Nak Counter reload state machine to WaitForListHead (see Nak Count Reload Control).
C	The host controller cancels the sleep timer (*AsynchronousTraversalSleepTimer*).

65.4.3.8.4.2 Async Sched Not Active

This is the initial state of the traversal state machine after a host controller reset. The traversal state machine does not leave this state when the *Asynchronous Schedule Enable* bit in the USB_USBCMD register is zero.

This state is entered from Async Sched Active or Async Sched Sleeping states when the end-of-micro-frame event is detected.

65.4.3.8.4.3 Async Sched Active

This state is entered from the Async Sched Not Active state when the periodic schedule is not active. It is also entered from the Async Sched Sleeping states when the *AsyncrhonousTraversalSleepTimer* expires. On every transition into this state, the host controller sets the *Reclamation* bit in the USB_USBSTS register to one.

While in this state, the host controller continually traverses the asynchronous schedule until either the end of micro-frame or an empty list condition is detected.

65.4.3.8.4.4 Async Sched Sleeping

The state is entered from the Async Sched Active state when a schedule empty condition is detected. On entry to this state, the host controller sets the *AsynchronousTraversalSleepTimer* to *AsyncSchedSleepTime*.

65.4.3.8.4.5 Example Derivation for *AsyncSchedSleepTime*

The derivation is based on analysis of what work the host controller could be doing next.

It assumes the host controller does not keep any state about what work is possibly pending in the asynchronous schedule. The schedule could contain any mix of the possible combinations of high- full- or low-speed control and bulk requests.

The table below summarizes some of the typical 'next transactions' that could be in the schedule, and the amount of time (for example *footprint*, or *wall clock*) the transaction takes to complete.

Table 65-41. Typical Low-/Full-speed Transaction Times

Transaction Attributes		Footprint (time)	Description
Speed	HS	11.9 ms	Maximum foot print for a worst-case, full-sized bulk data transaction.
Size	512	9.45 ms	Maximum footprint for an approximate best-case, full-sized bulk data transaction.
Type	Bulk		
Speed	FS	~50 ms	Approximate typical for full-sized bulk data. An 8-byte low-speed is about 2x, or between 90 and 100 ms.
Size	64		
Type	Bulk		
Speed	FS	~12 ms	Approximate typical for 8-byte bulk/control (that is setup)
Size	8		
Type	Cntrl		

A *AsyncSchedSleepTime* value of 10 µs provides a reasonable relaxation of the system memory load and still provides a good level of service for the various transfer types and payload sizes. For example, say we detect an empty list after issuing a start-split for a 64-byte full-speed bulk request. Assuming this is the only thing in the list, the host controller gets the results of the full-speed transaction from the hub during the fifth complete-split request. If the full-speed transaction was an IN and it nak'd, the 10 µs sleep period would allow the host controller to get the NAK results on the first complete-split.

65.4.3.8.5 Asynchronous schedule traversal: *Start Event*

Once the HC has *idled* itself through the empty schedule detection (Section 0), it will naturally *activate* and begin processing from the Periodic Schedule at the beginning of each micro-frame.

In addition, it may have idled itself early in a micro-frame. When this occurs (idles early in the micro-frame) the HC must occasionally *re-activate* during the micro-frame and traverse the asynchronous schedule to determine whether any progress can be made. The requirements and method for this restart are described in Restarting Asynchronous Schedule Before EOF . Asynchronous schedule *Start Event*s are defined to be:

- Whenever the host controller transitions from the periodic schedule to the asynchronous schedule. If the periodic schedule is disabled and the asynchronous schedule is enabled, then the beginning of the micro-frame is equivalent to the transition from the periodic schedule, or
- The asynchronous schedule traversal restarts from a sleeping state (see Restarting Asynchronous Schedule Before EOF).

65.4.3.8.6 Reclamation Status Bit (USBSTS Register)

The operation of the empty asynchronous schedule detection feature (see Empty Asynchronous Schedule Detection) depends on the proper management of the *Reclamation* bit in the USB_USBSTS register.

The host controller tests for an empty schedule just after it fetches a new queue head while traversing the asynchronous schedule (see Fetch Queue Head).

It is required that the host controller sets the *Reclamation* bit to one whenever an asynchronous schedule traversal *Start Event*, as documented inAsynchronous schedule traversal: *Start Event*, occurs. The *Reclamation* bit is also set to one whenever the host controller executes a transaction while traversing the asynchronous schedule (see Execute Transaction). The host controller sets the *Reclamation* bit to zero whenever it finds a queue head with its *H-bit* set to one. Software should only set a queue head's *H-bit* if the queue head is in the asynchronous schedule. If software sets the *H-bit* in an interrupt queue head to one, the resulting behavior is undefined. The host controller may set the *Reclamation* bit to zero when executing from the periodic schedule.

65.4.3.9 Operational Model for Nak Counter

This section describes the operational model for the *NakCnt* field defined in a queue head.

See Queue Head Initialization for more information. Software should not use this feature for interrupt queue heads. This rule is not required to be enforced by the host controller.

USB protocol has built-in flow control through the Nak response by a device. There are several scenarios, beyond the Ping feature, where an endpoint may naturally Nak or Nyet the majority of the time. An example is the host controller management of the split transaction protocol for control and bulk endpoints. All bulk endpoints (High- or Full-speed) are serviced through the same asynchronous schedule. The time between the *Start-split* transaction and the first *Complete-split* transaction could be very short (that is like when the endpoint is the only one in the asynchronous schedule). The hub NYETs (effectively Naks) the *Complete-split* transaction until the classic transaction is complete. This could result in the host controller thrashing memory, repeatedly fetching the queue head and executing the transaction to the Hub, which does not complete until after the transaction on the classic bus completes.

The two component fields in a queue head to support the throttling feature: a counter field (*NakCnt*), and a counter reload field (*RL*). *NakCnt* is used by the host controller as one of the criteria to determine whether or not to execute a transaction to the endpoint. The two operational modes associated with this counter:

- Not Used- This mode is set when the *RL* field is zero. The host controller ignores the *NakCnt* field for any execution of transactions through a queue head with an *RL* field of zero. Software must use this selection for interrupt endpoints.
- Nak Throttle Mode- This mode is selected when the *RL* field is non-zero. In this mode, the value in the *NakCnt* field represents the maximum number of Nak or Nyet responses the host controller tolerates on each endpoint. In this mode, the HC decrements the *NakCnt* field based on the token/handshake criteria listed in the table below. The host controller must reload *NakCnt* when the endpoint successfully moves data (for example, policy to reward device for moving data).

The following table describes the NakCnt field adjustment rules.

Table 65-42. NakCnt Field Adjustment Rules

Token	Handshake	
	Handshake NAK	NYET
IN/PING	decrement *NakCnt*	N/A (protocol error)
OUT	decrement *NakCnt*	No Action[1] Start
Split	decrement *NakCnt*	N/A (protocol error)
Complete Split	No Action	Decrement *NakCnt*

1. Recommended behavior on this response is to reload *NakCnt*

In summary, system software enables the counter by setting the reload field (*RL*) to a non-zero value. The host controller may execute a transaction if *NakCnt* is non-zero. The host controller does not execute a transaction if *NakCnt* is zero. The reload mechanism is described in detail in Nak Count Reload Control .

NOTE

When all queue heads in the Asynchronous Schedule either exhausts all transfers or all NakCnt's go to zero, then the host controller detects an empty Asynchronous Schedule and idle schedule traversal (see Empty Asynchronous Schedule Detection).

Any time the host controller begins a new traversal of the Asynchronous Schedule, a *Start Event* is assumed, see Asynchronous schedule traversal: *Start Event*. Every time a Start-Event occurs, the Nak Count reload procedure is enabled.

65.4.3.9.1 Nak Count Reload Control

When the host controller reaches the *Execute Transaction* state for a queue head (meaning that it has an active operational state), it checks to determine whether the *NakCnt* field should be reloaded from *RL* (see Execute Transaction). If the answer is yes, then *RL* is copied into *NakCnt*. After the reload or if the reload is not active, the host controller evaluates whether to execute the transaction.

The host controller must reload nak counters (*NakCnt* see Table 65-25) in queue heads during the first pass through the reclamation list after an asynchronous schedule Start Event (see Asynchronous schedule traversal: *Start Event* for the definition of the Start Event). The Asynchronous Schedule should have at most one queue head marked as the head (see Figure 65-15).

The following figure illustrates an example state machine that satisfies the operational requirements of the host controller detecting the first pass through the Asynchronous Schedule. This state machine is maintained internal to the host controller and is only used to gate reloading of the nak counter during the queue head traversal state: Execute Transaction (see the figure below). The host controller does not perform the nak counter reload operation if the RL field (see Table 65-25) is set to zero.

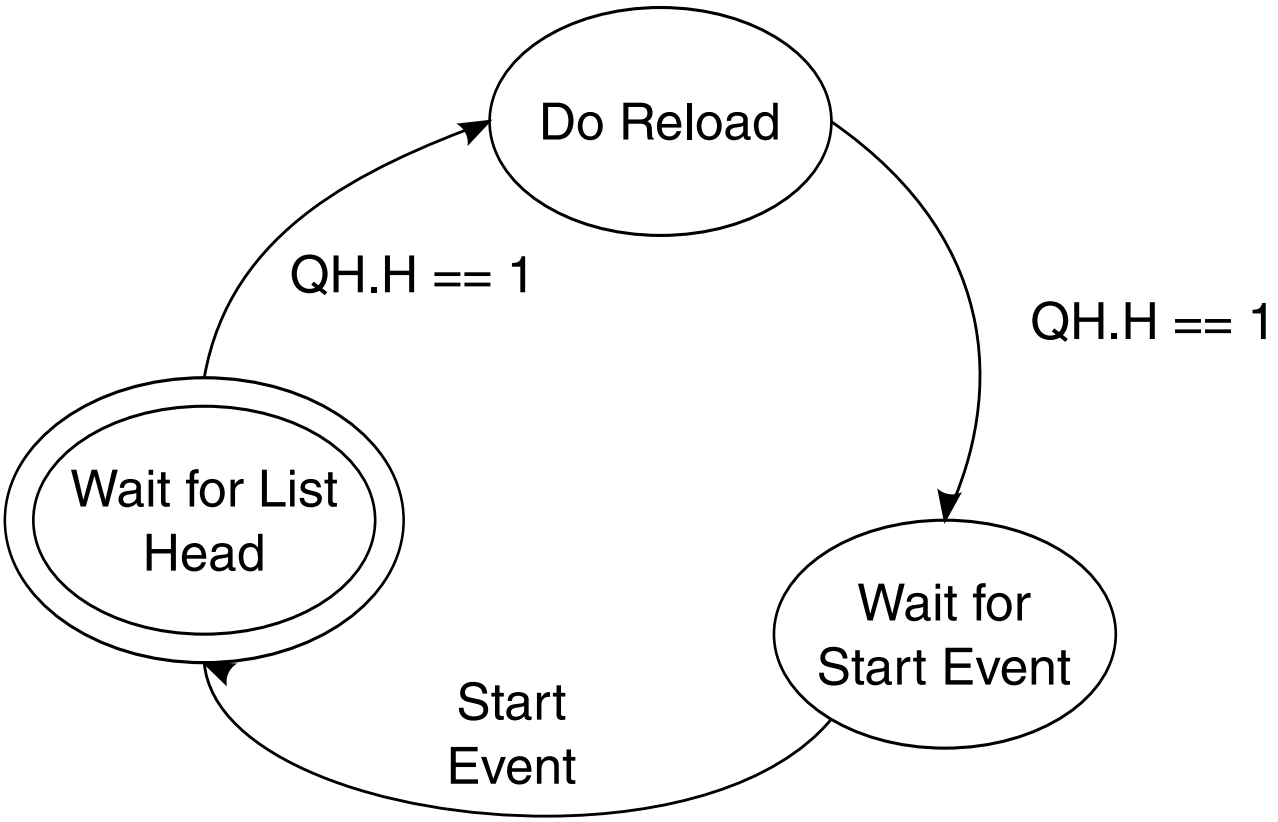


Figure 65-17. Example HC State Machine for Controlling Nak Counter Reloads

65.4.3.9.1.1 Wait for List Head

This is the initial state.

The state machine enters this state from Wait for Start Event when a start event as defined in Asynchronous schedule traversal: *Start Event*occurs.

The purpose of this state is to wait for the first observation of the head of the Asynchronous Schedule.

This occurs when the host controller fetches a queue head whose *H-bit* is set to one.

65.4.3.9.1.2 Do Reload

This state is entered from the Wait for List Head state when the host controller fetches a queue head with the *H-bit* set to one. While in this state, the host controller performs nak counter reloads for every queue head visited that has a non-zero nak reload value (*RL*) field.

65.4.3.9.1.3 Wait for Start Event

This state is entered from the *Do Reload* state when a queue head with the *H-bit* set to one is fetched. While in this state, the host controller does not perform nak counter reloads.

65.4.3.10 Managing Control/Bulk/Interrupt Transfers through Queue Heads

This section presents an overview of how the host controller interacts with queuing data structures.

Queue heads use the Queue Element Transfer Descriptor (qTD) structure. One queue head is used to manage the data stream for one endpoint. The queue head structure contains static endpoint characteristics and capabilities. It also contains a working area from where individual bus transactions for an endpoint are executed (see Overlay area defined in Table 65-25). Each qTD represents one or more bus transactions, which is defined in the context of this specification as a *transfer*.

The general processing model for the host controller's use of a queue head is simple:

- read a queue head,
- execute a transaction from the overlay area,
- write back the results of the transaction to the overlay area,
- move to the next queue head.

If the host controller encounters errors during a transaction, the host controller sets one (or more) of the error reporting bits in the queue head's *Status* field. The *Status* field accumulates all errors encountered during the execution of a qTD (for example, the error bits in the queue head *Status* field are 'sticky' until the transfer (qTD) has completed). This state is always written back to the source qTD when the transfer is complete. On transfer (for example, buffer or halt conditions) boundaries, the host controller must auto-advance (without software intervention) to the next qTD. Additionally, the hardware must be able to halt the queue so no additional bus transactions occurs for the endpoint and the host controller does not advance the queue.

An example host controller operational state machine of a queue head traversal is illustrated in the following figure. This state machine is a model for how a host controller should traverse a queue head. The host controller must be able to advance the queue from the *Fetch QH* state in order to avoid all hardware/software race conditions. This simple mechanism allows software to simply link qTDs to the queue head and *activate* them, then the host controller always *find* them if/when they are reachable. The figure below illustrates the Host Controller Queue Head Traversal State Machine.

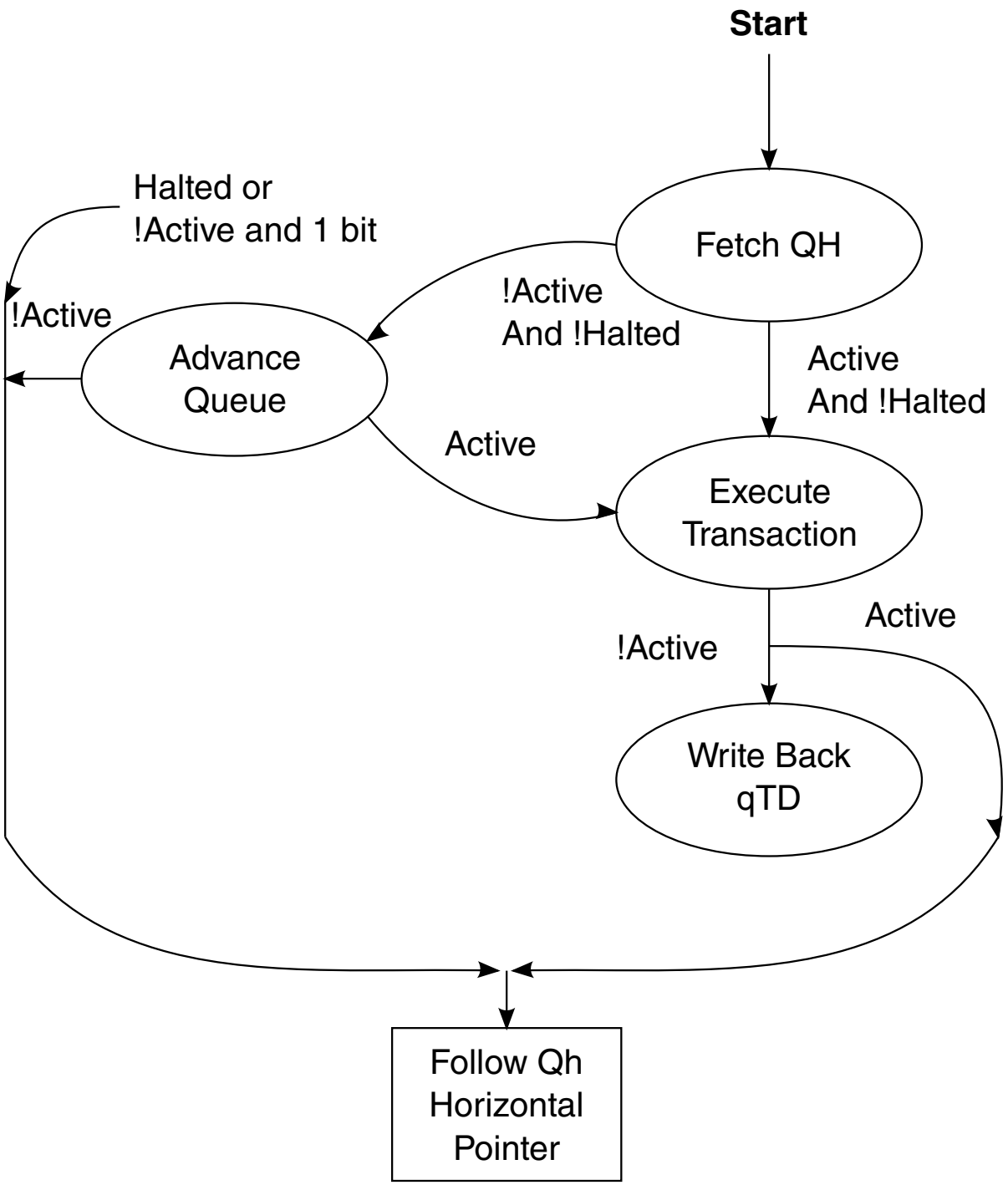


Figure 65-18. Host Controller Queue Head Traversal State Machine

This traversal state machine applies to all queue heads, regardless of transfer type or whether split transactions are required. The following sections describe each state. Each state description describes the entry criteria. The Execute Transaction state (see Execute

Transaction) describes the basic requirements for all endpoints. Split Transactions for Asynchronous Transfers and Split Transaction Interrupt describe details of the required extensions to the Execute Transaction state for endpoints requiring split transactions.

NOTE

Prior to software placing a queue head into either the periodic or asynchronous list, software must ensure the queue head is properly initialized. Minimally, the queue head should be initialized to the following (see Section Queue Head for layout of a queue head):

Valid static endpoint state.

- For the very first use of a queue head, software may zero-out the queue head transfer overlay, then set the *Next qTD Pointer* field value to reference a valid qTD.

65.4.3.10.1 Fetch Queue Head

A queue head can be referenced from the physical address stored in the ASYNCLISTADDR Register (see Next Asynch. Address (USBC_*n*_ASYNCLISTADDR))/Endpoint List Address (USBC_*n*_ENDPTLISTADDR) Additionally, it may be referenced from the *Next LinkPointer* field of an iTD, siTD, FSTN or another Queue Head. If the referencing link pointer has the *Typ* field set to indicate a queue head, it is assumed to reference a queue head structure as defined in Table 65-25.

While in this state, the host controller performs operations to implement empty schedule detection (see Empty Asynchronous Schedule Detection) and Nak Counter reloads (see Operational Model for Nak Counter). After the queue head has been fetched, the host controller conducts the following queries for empty schedule detection:

- If queue head is not an interrupt queue head (that is *S-mask* is zero), and
- The *H-bit* is one, and
- The *Reclamation* bit in the USBSTS register is zero.

When these criteria are met, the host controller stops traversing the asynchronous list (as described in Empty Asynchronous Schedule Detection). When the criteria are not met, the host controller continues schedule traversal. If the queue head is not an interrupt and the *H-bit* is one and the *Reclamation* bit is one, then the host controller sets the *Reclamation* bit in the USBSTS register to zero before completing this state. The operations for reloading of the Nak Counter are described in detail in Operational Model for Nak Counter.

This state is complete when the queue head has been read on-chip.

65.4.3.10.2 Advance Queue

To advance the queue, the host controller must find the next qTD, adjust pointers, perform the overlay and write back the results to the queue head.

This state is entered from the FetchQHD state if the overlay *Active* and *Halt* bits are set to zero. On entry to this state, the host controller determines which next pointer to use to fetch a qTD, fetches a qTD and determines whether or not to perform an overlay.

NOTE

If the *I-bit* is one and the *Active* bit is zero, the host controller immediately skips processing of this queue head, exits this state and uses the horizontal pointer to the next schedule data structure. If the field *Bytes to Transfer* is not zero and the *T-bit* in the *Alternate Next qTD Pointer* is set to zero, then the host controller uses the *Alternate Next qTD Pointer*. Otherwise, the host controller uses the *NextqTD Pointer*. If *NextqTD Pointer's T-bit* is set to one, then the host controller exits this state and uses the horizontal pointer to the next schedule data structure.

Using the selected pointer the host controller fetches the referenced qTD. If the fetched qTD has its *Active* bit set to one, the host controller moves the pointer value used to reach the qTD (*Next* or *Alternate Next*) to the *Current qTD Pointer* field, then performs the overlay. If the fetched qTD has its *Active* bit set to zero, the host controller aborts the queue advance and follows the queue head's horizontal pointer to the next schedule data structure.

The host controller performs the overlay based on the following rules:

- The value of the data toggle (*dt*) field in the overlay area depends on the value of the *data toggle control* (*dtc*) bit (see Table 65-27).
- If the *EPS* field indicates the endpoint is a high-speed endpoint, the *Ping* state field is preserved by the host controller. The value of this field is not changed as a result of the overlay.
- *C-prog-mask* field is set to zero (field from incoming qTD is ignored, as is the current contents of the overlay area).
- *Frame Tag* field is set to zero (field from incoming qTD is ignored, as is the current contents of the overlay area).
- *NakCnt* field in the overlay area is loaded from the *RL* field in the queue head's Static Endpoint State.
- All other areas of the overlay are set by the incoming qTD.

The host controller exits this state when it has committed the write to the queue head.

65.4.3.10.3 Execute Transaction

The host controller enters this state from the Fetch Queue Head state only if the *Active* bit in *Status* field of the queue head is set to one.

On entry to this state, the host controller executes a few pre-operations, then checks some pre-condition criteria before committing to executing a transaction for the queue head.

The pre-operations performed and the pre-condition criteria depend on whether the queue head is an interrupt endpoint. The host controller can determine that a queue head is an interrupt queue head when the queue head's *S-mask* field contains a non-zero value. It is the responsibility of software to ensure the *S-mask* field is appropriately initialized based on the transfer type. There are other criteria that must be met if the *EPS* field indicates that the endpoint is a low- or full-speed endpoint, see Split Transactions for Asynchronous Transfers and Split Transaction Interrupt .

65.4.3.10.3.1 Interrupt Transfer Pre-condition Criteria

If the queue head is for an interrupt endpoint (for example, non-zero *S-mask* field), then the FRINDEX[2:0] field must identify a bit in the *S-mask* field that has one in it.

For example, an *S-mask* value of 00100000b would evaluate to true only when FRINDEX[2:0] is equal to 101b. If this condition is met then the host controller considers this queue head for a transaction.

65.4.3.10.3.2 Asynchronous Transfer Pre-operations and Pre-condition Criteria

If the queue head is not for an interrupt endpoint (for example, zero *S-mask* field), then the host controller performs one pre-operation and then evaluates one pre-condition criteria.

The pre-operation is:

Checks the Nak counter reload state (Operational Model for Nak Counter). It may be necessary for the host controller to reload the Nak Counter field. The reload is performed at this time.

The pre-condition evaluated is:

- Whether or not the *NakCnt* field has been reloaded, the host controller checks the value of the *NakCnt* field in the queue head. If *NakCnt* is non-zero, or if the *Reload Nak Counter* field is zero, then the host controller considers this queue head for a transaction.

65.4.3.10.3.3 Transfer Type Independent Pre-operations

Regardless of the transfer type, the host controller always performs at least one pre-operation and evaluates one pre-condition. The pre-operation is:

- A host controller internal transaction (down) counter *qHTransactionCounter* is loaded from the queue head's *Mult* field. A host controller implementation is allowed to ignore this for queue heads on the asynchronous list. It is mandatory for interrupt queue heads. Software should ensure that the *Mult* field is set appropriately for the transfer type.

The pre-conditions evaluated are:

- The host controller determines whether there is enough time in the micro-frame to complete this transaction (see Transaction Fit - A Best-Fit Approximation Algorithm for an example evaluation method). If there is not enough time to complete the transaction, the host controller exits this state.
- If the value of *qHTransactionCounter* for an interrupt endpoint is zero, then the host controller exits this state.

When the pre-operations are complete and pre-conditions are met, the host controller sets the *Reclamation* bit in the USBSTS register to one and then begins executing one or more transactions using the endpoint information in the queue head. The host controller iterates *qHTransactionCounter* times in this state executing transactions. After each transaction is executed, *qHTransactionCounter* is decremented by one. The host controller exits this state when one of the following events occurs:

- The *qHTransactionCounter* decrements to zero, or
- The endpoint responds to the transaction with any handshake other than an ACK,[4] or
- The transaction experiences a transaction error, or
- The *Active* bit in the queue head goes to zero, or
- There is not enough time in the micro-frame left to execute the next transaction(see Transaction Fit - A Best-Fit Approximation Algorithm) for example method for implementing the frame boundary test).

NOTE

For a high-bandwidth interrupt OUT endpoint, the host controller may optionally immediately retry the transaction if it fails.

The results of each transaction is recorded in the on-chip overlay area. If data was successfully moved during the transaction, the transfer state in the overlay area is advanced. To advance queue head's transfer state, the *Total Bytes to Transfer* field is decremented by the number of bytes moved in the transaction, the data toggle bit (*dt*) is toggled, the current page offset is advanced to the next appropriate value (for example,

advanced by the number of bytes successfully moved), and the *C_Page* field is updated to the appropriate value (if necessary). See Buffer Pointer List Use for Data Streaming with qTDs .

NOTE

The *Total Bytes To Transfer* field may be zero when all the other criteria for executing a transaction are met. When this occurs, the host controller executes zero-length transaction to the endpoint. If the *PID_Code* field indicates an IN transaction and the device delivers data, the host controller detects a packet babble condition, set the *babble* and *halted* bits in the *Status* field, set the *Active* bit to zero, write back the results to the source qTD, then exit this state.

In the event an IN token receives a data PID mismatch response, the host controller must ignore the received data (for example not advance the transfer state for the bytes received). Additionally, if the endpoint is an interrupt IN, then the host controller must record that the transaction occurred (for example, decrement *qHTransactionCounter*). It is recommended (but not required) the host controller continue executing transactions for this endpoint if the resultant value of *qHTransactionCounter* is greater than one.

If the response to the IN bus transaction is a Nak (or Nyet) and *RL* is non-zero, *NakCnt* is decremented by one. If *RL* is zero, then no write-back by the host controller is required (for a transaction receiving a Nak or Nyet response and the value of *CErr* did not change). Software should set the *RL* field to zero if the queue head is an interrupt endpoint. Host controller hardware is not required to enforce this rule or operation.

After the transaction has finished and the host controller has completed the post processing of the results (advancing the transfer state and possibly *NakCnt*, the host controller writes back the results of the transaction to the queue head's overlay area in main memory).

The number of bytes moved during an IN transaction depends on how much data the device endpoint delivers. The maximum number of bytes a device can send is *MaximumPacket Size*. The number of bytes moved during an OUT transaction is either *Maximum Packet Length* bytes or *Total Bytes to Transfer*, whichever is less.

If there was a transaction error during the transaction, the transfer state (as defined above) is not advanced by the host controller. The *CErr* field is decremented by one and the status field is updated to reflect the type of error observed. Transaction errors are summarized in Transaction Error .

The following events causes the host controller to clear the *Active* bit in the queue head's overlay status field. When the *Active* bit transitions from one to zero, the transfer in the overlay is considered complete. The reason for the transfer completion (clearing the *Active* bit) determines the next state.

- *CErr* field decrements to zero. When this occurs the *Halted* bit is set to one and *Active* is set to zero. This results in the hardware not advancing the queue and the pipe halts. Software must intercede to recover.
- The device responds to the transaction with a STALL PID. When this occurs, the *Halted* bit is set to one and the *Active* bit is set to zero. This results in the hardware not advancing the queue and the pipe halts. Software must intercede to recover.
- The *Total Bytes to Transfer* field is zero after the transaction completes.
  - For a zero length transaction, it was zero before the transaction was started. When this condition occurs, the *Active* bit is set to zero.
- The PID code is an IN, and the number of bytes moved during the transaction is less than the *Maximum Packet Length*. When this occurs, the *Active* bit is set to zero and a short packet condition exists. The short-packet condition is detected during the Advance Queue state. Refer to Split Transactions for additional rules for managing low- and full-speed transactions.

With the exception of a NAK response (when *RL* field is zero), the host controller always writes the results of the transaction back to the overlay area in main memory. This includes when the transfer completes. For a high-speed endpoint, the queue head information written back includes minimally the following fields: The *PID Code* field indicates an IN and the device sends more than the expected number of bytes (for example *Maximum Packet Length* or *Total Bytes to Transfer* bytes, whichever is less) (for example a packet babble). This results in the host controller setting the *Halted* bit to one.

- NakCnt, dt, Total Bytes to Transfer, C_Page, Status, CERR, and Current Offset

For a low- or full-speed device the queue head information written back also includes the fields:

- C-prog-mask, FrameTag and S-bytes.

The duration of this state depends on the time it takes to complete the transaction(s) and the status write to the overlay is committed.

65.4.3.10.3.4 Halting a Queue Head

A halted endpoint is defined only for the transfer types that are managed through queue heads (control, bulk and interrupt).

The following events indicate that the endpoint has reached a condition where no more activity can occur without intervention from the driver:

- An endpoint may return a STALL handshake during a transaction,
- A transaction had three consecutive error conditions, or
- A Packet Babble error occurs on the endpoint.

When any of these events occur (for a queue head) the Host Controller halts the queue head and set the USBERRINT status bit in the USB_n_USBSTS register to one. To halt the queue head, the *Active* bit is set to zero and the *Halted* bit is set to one. There may be other error status bits that are set when a queue is halted. The host controller always writes back the overlay area to the source qTD when the transfer is complete, regardless of the reason (normal completion, short packet or halt). The host controller does not advance the transfer state on a transaction that results in a *Halt* condition (for example no updates necessary for *Total Bytes to Transfer*, *C_Page*, *Current Offset,* and *dt*). The host controller must update *CErr* as appropriate. When a queue head is halted, the *USB Error Interrupt* bit in the USB_n_USBSTS register is set to one. If the *USB Error Interrupt Enable* bit in the USB_n_USBINTR register is set to one, a hardware interrupt is generated at the next interrupt threshold.

65.4.3.10.3.5 Asynchronous Schedule Park Mode

Asynchronous Schedule Park mode is a special execution mode that can be enabled by system software, where the host controller is permitted to execute more than one bus transaction from a high-speed queue head in the Asynchronous schedule before continuing horizontal traversal of the Asynchronous schedule.

This feature has no effect on queue heads or other data structures in the Periodic schedule. This feature is similar in intent as the *Mult* feature that is used in the Periodic schedule. Where-as the *Mult* feature is a characteristic that is tunable for each endpoint; park-mode is a policy that is applied to all high-speed queue heads in the asynchronous schedule. It is essentially the specification of an iterator for consecutive bus transactions to the same endpoint. All of the rules for managing bus transactions and the results of those as defined in Execute Transaction apply. This feature merely specifies how many consecutive times the host controller is permitted to execute from the same queue head before moving to the next queue head in the Asynchronous List. This feature should allow the host controller to attain better bus utilization for those devices that are capable of moving data at maximum rate, while at the same time providing a fair service to all endpoints.

A host controller exports its capability to support this feature to system software by setting the *Asynchronous Schedule Park Capability* bit in the USB_n_HCCPARAMs register to one. This information keys system software that the *Asynchronous Schedule Park Mode Enable* and *Asynchronous Schedule Park Mode Count* fields in the USB_n_USBCMD register are modifiable. System software enables the feature by writing a one to the *Asynchronous Schedule Park Mode Enable* bit.

When park-mode is not enabled (for example *Asynchronous Schedule Park Mode Enable* bit in the USB_n_USBCMD register is zero), the host controller must not execute more than one bus transaction per high-speed queue head, per traversal of the asynchronous schedule. When park-mode is enabled, the host controller must not apply the feature to a queue head whose *EPS* field indicates a Low/Full-speed device (for example only one bus transaction is allowed from each Low/Full-speed queue head per traversal of the asynchronous schedule). Park-mode may only be applied to queue heads in the Asynchronous schedule whose *EPS* field indicates that it is a high-speed device.

The host controller must apply park mode to queue heads whose *EPS* field indicates a high-speed endpoint. The maximum number of consecutive bus transactions a host controller may execute on a high-speed queue head is determined by the value in the *Asynchronous Schedule Park Mode Count* field in the USB_n_USBCMD register. Software must not set *Asynchronous Schedule Park Mode Enable* bit to one and also set *Asynchronous Schedule Park Mode Count* field to zero. The resulting behavior is not defined. An example behavioral example describes the operational requirements for the host controller implementing park-mode. This feature does not affect how the host controller handles the bus transaction as defined in Execute Transaction . It only effects how many consecutive bus transactions for the current queue head can be executed. All boundary conditions, error detection and reporting applies as usual. This feature is similar in concept to the use of the *Mult* field for high-bandwidth Interrupt for queue heads in the Periodic Schedule.

The host controller effectively loads an internal down-counter *PM-Count* from *Asynchronous Schedule Park Mode Count* when *Asyncrhonous Schedule Park Mode Enable* bit is one, and a high-speed queue head is first fetched and meets all the criteria for executing a bus transaction. After the bus transaction, *PM-Count* is decremented. The host controller may continue to execute bus transactions from the current queue head until *PM-Count* goes to zero, an error is detected, the buffer for the current transfer is exhausted or the endpoint responds with a flow-control or STALL handshake.

The following table summarizes the responses that effect whether the host controller continues with another bus transaction for the current queue head.

Table 65-43. Actions for Park Mode, based on Endpoint Response and Residual Transfer State

PID	Endpoint Response	Transfer State after Transaction		Action
		PM-Count	Bytes to Transfer	
IN	DATA[0,1] w/Maximum Packet sized data	Not zero	Not Zero	Allowed to perform another bus transaction.[1, 2]
		Not zero	Zero	Retire qTD and move to next QH
		Zero	Don't care	Move to next QH.

Table continues on the next page...

Table 65-43. Actions for Park Mode, based on Endpoint Response and Residual Transfer State (continued)

	DATA[0,1] w/short packet	Don't care	Don't care	Retire qTD and move to next QH.
	NAK	Don't care	Don't care	Move to next QH.
	STALL, XactErr	Don't care	Don't care	Move to next QH.
OUT	ACK	Not zero	Not Zero	Allowed to perform another bus transaction.[2]
		Not zero	Zero	Retire qTD and move to next QH
		Zero	Don't' care	Move to next QH.
	NYET, NAK	Don't care	Don't care	Move to next QH.
	STALL, XactErr	Don't care	Don't care	Move to next QH
PING	ACK	Not Zero	Not Zero	Allowed to perform another bus transaction. [2]
	NAK	Don't care	Don't care	Move to next QH
	STALL, XactErr	Don't care	Don't care	Move to next QH

1. The host controller may continue to execute bus transactions from the current high-speed queue head (if *PM-Count* is not equal to zero), if a PID mismatch is detected (for example expected DATA1 and received DATA0, or visa-versa).
2. This specification does not *require* that the host controller execute another bus transaction when *PM-Count* is non-zero. Implementations are encouraged to make appropriate complexity and performance trade-offs.

65.4.3.10.4 Write Back qTD

This state is entered from the Execute Transaction state when the *Active* bit is set to zero.

The source data for the write-back is the transfer results area of the queue head overlay area (see Table 65-43).

The host controller uses the *Current qTD Pointer* field as the target address for the qTD.

The queue head transfer result area is written back to the transfer result area of the target qTD. This state is also referred to as: qTD retirement. The fields that must be written back to the source qTD include *Total Bytes to Transfer*, *Cerr*, and *Status*.

The duration of this state depends on when the qTD write-back is committed.

65.4.3.10.5 Follow Queue Head Horizontal Pointer

The host controller must use the horizontal pointer in the queue head to the next schedule data structure when any of the following conditions exist:

- If the *Active* bit is one on exit from the Execute Transaction state, or
- When the host controller exits the Write Back qTD state, or
- If the Advance Queue state fails to advance the queue because the target qTD is not active, or
- If the *Halted* bit is one on exit from the Fetch QH state.

There is no functional requirement that the host controller wait until the current transaction is complete before using the horizontal pointer to read the next linked data structure. However, it must wait until the current transaction is complete before executing the next data structure.

65.4.3.10.6 Buffer Pointer List Use for Data Streaming with qTDs

A qTD has an array of buffer pointers, which is used to reference the data buffer for a transfer. This specification requires that the buffer associated with the transfer be *virtually contiguous*.

This means: if the buffer spans more than one physical page, it must obey the following rules (the figure below illustrates an example):

- The first portion of the buffer must begin at some offset in a page and extend through the end of the page.
- The remaining buffer cannot be allocated in small chunks scattered around memory. For each 4 K chunk beyond the first page, each buffer portion matches to a full 4 K page. The final portion, which may only be large enough to occupy a portion of a page, must start at the top of the page and be contiguous within that page.

The buffer pointer list in the qTD is long enough to support a maximum transfer size of 20 K bytes. This case occurs when all five buffer pointers are used and the first offset is zero. A qTD handles a 16 Kbyte buffer with any starting buffer alignment.

The host controller uses the field *C_Page* field as an index value to determine which buffer pointer in the list should be used to start the current transaction. The host controller uses a different buffer pointer for each physical page of the buffer. This is always true, even if the buffer is physically contiguous.

The host controller must detect when the current transaction spans a page boundary and automatically move to the next available buffer pointer in the page pointer list. The next available pointer is reached by incrementing *C_Page* and pulling the next page pointer from the list. Software must ensure there are sufficient buffer pointers to move the amount of data specified in the *Bytes to Transfer* field.

The following figure illustrates a nominal example of how System software would initialize the buffer pointers list and the *C_Page* field for a transfer size of 16383 bytes. *C_Page* is set to zero. The upper 20-bits of Page 0 references the start of the physical page. *Current Offset* (the lower 12-bits of queue head Dword 7) holds the offset in the page for example 2049 (for example 4096-2047). The remaining page pointers are set to reference the beginning of each subsequent 4 K page.

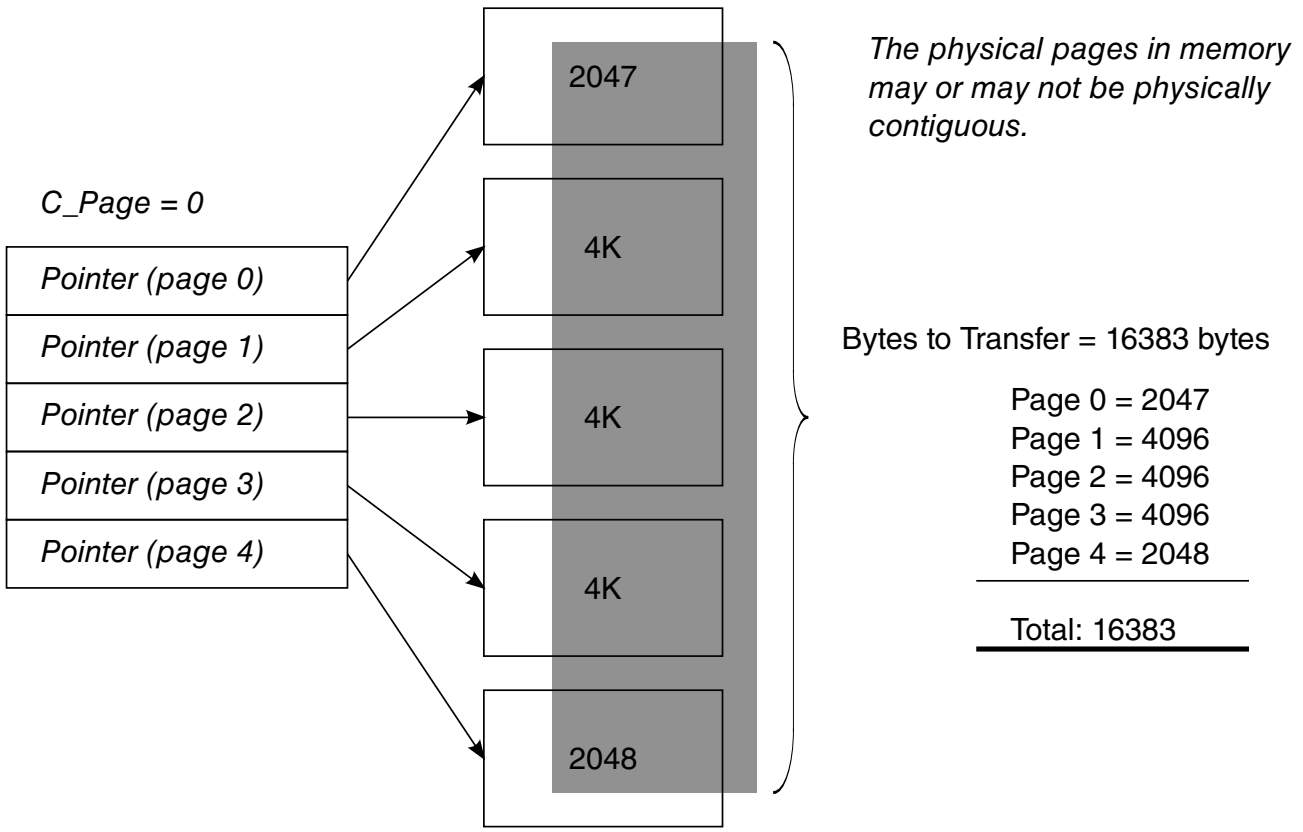


Figure 65-19. Example Mapping of qTD Buffer Pointers to Buffer Pages

For the first transaction on the qTD (assuming a 512-byte transaction), the host controller uses the first buffer pointer (page 0 because *C_Page* is set to zero) and concatenates the *Current Offset* field. The 512 bytes are moved during the transaction, the *Current Offset* and *Total Bytes to Transfer* are adjusted by 512 and written back to the queue head working area.

During the 4th transaction, the host controller needs 511 bytes in page 0 and one byte in page 1. The host controller increments *C_Page* (to 1) and use the page 1 pointer to move the final byte of the transaction. After the 4th transaction, the active page pointer is the page 1 pointer and *Current Offset* has rolled to one, and both are written back to the overlay area. The transactions continue for the rest of the buffer, with the host controller automatically moving to the next page pointer (that is *C_Page*) when necessary. The three conditions for how the host controller handles *C_Page*:

- The current transaction does not span a page boundary. The value of *C_Page* is not adjusted by the host controller.

- The current transaction does span a page boundary. The host controller must detect the page cross condition and advance to the next buffer while streaming data to/from the USB.
- The current transaction completes on a page boundary (that is the last byte moved for the current transaction is the last byte in the page for the current page pointer). The host controller must increment *C_Page* before writing back status for the transaction.

NOTE

The only valid adjustment the host controller may make to *C_Page* is to increment by one.

65.4.3.10.7 Adding Interrupt Queue Heads to the Periodic Schedule

The link path(s) from the periodic frame list to a queue head establishes in which frames a transaction can be executed for the queue head. Queue heads are linked into the periodic schedule so they are polled at the appropriate rate.

System software sets a bit in a queue head's *S-Mask* to indicate which micro-frame within 1 msec period a transaction should be executed for the queue head. Software must ensure that all queue heads in the periodic schedule have *S-Mask* set to a non-zero value. An *S-mask* with zero value in the context of the periodic schedule yields undefined results.

If the desired poll rate is greater than one frame, system software can use a combination of queue head linking and *S-Mask* values to spread interrupts of equal poll rates through the schedule so that the periodic bandwidth is allocated and managed in the most efficient manner possible. Some examples are illustrated in the following table.

Table 65-44. Example Periodic Reference Patterns for Interrupt Transfers with 2ms Poll Rate

Frame # Reference Sequence	Description
0, 2, 4, 6, 8, and so on *S-Mask* = 01h	A queue head for the *bInterval* of 2 msec (16 micro-frames) is linked into the periodic schedule so that it is reachable from the periodic frame list locations indicated in the previous column. In addition, the *S-Mask* field in the queue head is set to 01h, indicating that the transaction for the endpoint should be executed on the bus during micro-frame 0 of the frame.
0, 2, 4, 6, 8, and so on *S-Mask* = 02h	Another example of a queue head with a *bInterval* of 2 msec is linked into the periodic frame list at exactly the same interval as the previous example. However, the *S-Mask* is set to 02h indicating that the transaction for the endpoint should be executed on the bus during micro-frame 1 of the frame.

65.4.3.10.8 Managing Transfer Complete Interrupts from Queue Heads

The host controller sets an interrupt to be signaled at the next interrupt threshold when the completed transfer (qTD) has an *Interrupt on Complete* (*IOC*) bit set to one, or whenever a transfer (qTD) completes with a short packet.

If system software needs multiple qTDs to complete a client request (that is like a control transfer) the intermediate qTDs do not require interrupts. System software may only need a single interrupt to notify it that the complete buffer has been transferred. System software may set IOC's to occur more frequently. A motivation for this may be that it wants early notification so that interface data structures can be re-used in a timely manner.

65.4.3.11 Ping Control

USB 2.0 defines an addition to the protocol for high-speed devices called Ping. Ping is required for all USB 2.0 High-speed bulk and control endpoints.

Ping is not allowed for a split-transaction stream. This extension to the protocol eliminates the bad side-effects of Naking OUT endpoints. The *Status* field has a *Ping State* bit, which the host controller uses to determine the *next* actual PID it uses in the next transaction to the endpoint (see the table below).

The Ping State bit is only managed by the host controller for queue heads that meet the following criteria:

- Queue head is not an interrupt and
- *EPS* field equals High-Speed and
- *PIDCode* field equals OUT

The following table illustrates the state transition table for the host controller's responsibility for maintaining the PING protocol. Refer to Chapter 8 in the USB Specification Revision 2.0 for detailed description on the Ping protocol.

Table 65-45. Ping Control State Transition Table

Event			
Current	Host	Device	Next
Do Ping	PING	Nak	Do Ping
Do Ping	PING	Ack	Do OUT
Do Ping	PING	XactErr [1]	Do Ping
Do Ping	PING	Stall	N/C [2] Do
OUT	OUT	Nak	Do Ping
Do OUT	OUT	Nyet	Do Ping

Table continues on the next page...

Table 65-45. Ping Control State Transition Table (continued)

Do OUT	OUT	Ack	Do OUT
Do OUT	OUT	XactErr[1]	Do Ping
Do OUT	OUT	Stall	N/C [2]

1. Transaction Error (XactErr) is any time the host misses the handshake.
2. No transition change required for the Ping State bit. The Stall handshake results in the endpoint being halted (for example Active set to zero and Halt set to one). Software intervention is required to restart queue. 3 A Nyet response to an OUT means that the device has accepted the data, but cannot receive any more at this time. Host must advance the transfer state and additionally, transition the Ping State bit to Do Ping. The Ping State bit has the following encoding:

Table 65-46. Ping State bit Encoding

Value	Meaning
0B	Do OUT The host controller uses an OUT PID during the next bus transaction to this endpoint.
1B	Do Ping The host controller uses a PING PID during the next bus transaction to this endpoint.

The defined ping protocol (see USB 2.0 Specification, Chapter 8) allows the host to be *imprecise* on the initialization of the ping protocol (that is start in *Do OUT* when we don't know whether there is space on the device or not). The host controller manages the *Ping State* bit. System software sets the initial value in the queue head when it initializes a queue head. The host controller preserves the *Ping State* bit across all queue advancements. This means that when a new qTD is written into the queue head overlay area, the previous value of the *Ping State* bit is preserved.

65.4.3.12 Split Transactions

USB 2.0 defines extensions to the bus protocol for managing USB 1.x data streams through USB 2.0 Hubs.

This section describes how the host controller uses the interface data structures to manage data streams with full- and low-speed devices, connected below USB 2.0 hub, utilizing the split transaction protocol.

Refer to USB 2.0 Specification for the complete definition of the split transaction protocol. Full- and Low-speed devices are enumerated identically as high-speed devices, but the transactions to the Full- and Low-speed endpoints use the split-transaction protocol on the high-speed bus. The split transaction protocol is an encapsulation of (or wrapper around) the Full- or Low-speed transaction. The high-speed wrapper portion of the protocol is addressed to the USB 2.0 Hub and Transaction Translator below which the Full- or Low-speed device is attached.

The EHCI interface uses dedicated data structures for managing full-speed isochronous data streams (see Split Transaction Isochronous Transfer Descriptor (siTD)). Control, Bulk and Interrupt are managed using the queuing data structures (see Queue Head). The interface data structures need to be programmed with the device address and the Transaction Translator number of the USB 2.0 Hub operating as the Low-/Full-speed host controller for this link. The following sections describe the details of how the host controller must process and manage the split transaction protocol.

65.4.3.12.1 Split Transactions for Asynchronous Transfers

A queue head in the asynchronous schedule with an *EPS* field indicating a full-or low-speed device indicates to the host controller that it must use split transactions to stream data for this queue head.

All full-speed bulk and full-, low-speed control are managed through queue heads in the asynchronous schedule.

Software must initialize the queue head with the appropriate device address and port number for the transaction translator that is serving as the full/low-speed host controller for the links connecting the endpoint. Software must also initialize the split transaction state bit (*SplitXState*) to Do-Start-Split. Finally, if the endpoint is a control endpoint, then system software must set the *Control Transfer Type* (*C*) bit in the queue head to one. If this is not a control transfer type endpoint, the *C* bit must be initialized by software to be zero. This information is used by the host controller to properly set the Endpoint Type (ET) field in the split transaction bus token. When the *C* bit is zero, the split transaction token's ET field is set to indicate a bulk endpoint. When the *C* bit is one, the split transaction token's ET field is set to indicate a control endpoint. Refer to Chapter 8 of USB Specification Revision 2.0 for details.

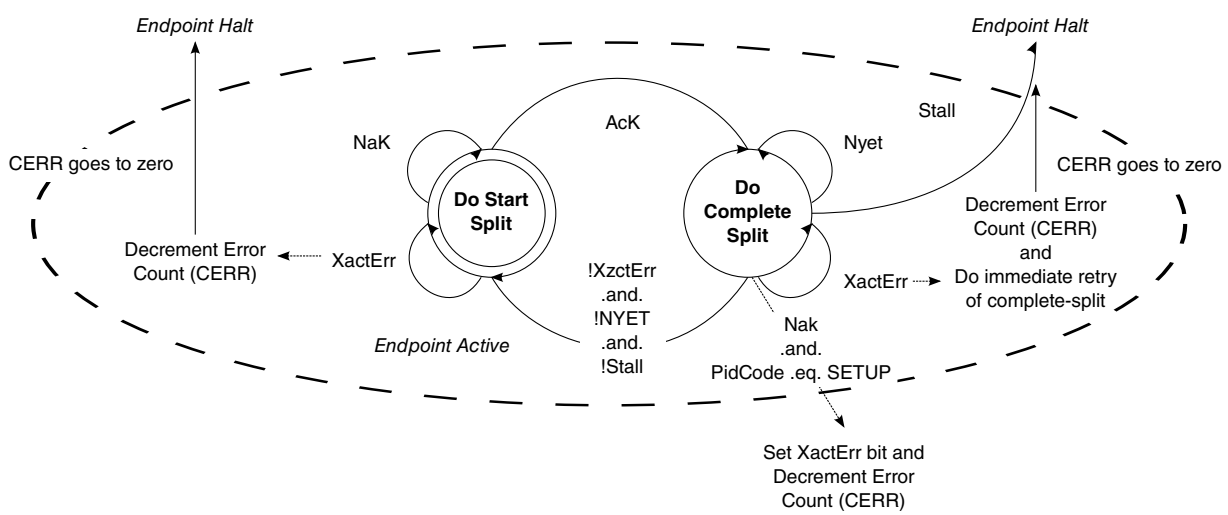


Figure 65-20. Host Controller Asynchronous Schedule Split-Transaction State Machine

65.4.3.12.1.1 Asynchronous - Do Start Split

This is the state which software must initialize a full- or low-speed asynchronous queue head. This state is entered from the Do Complete Split state only after a complete-split transaction receives a valid response from the transaction translator that is not a Nyet handshake.

For queue heads in this state, the host controller executes a start-split transaction to the appropriate transaction translator. If the bus transaction completes without an error and *PidCode* indicates an IN or OUT transaction, then the host controller reloads the error counter (*CErr*). If it is a successful bus transaction and the *PidCode* indicates a SETUP, the host controller does not reload the error counter. If the transaction translator responds with a Nak, the queue head is left in this state, and the host controller proceeds to the next queue head in the asynchronous schedule.

If the host controller times out the transaction (no response, or bad response) the host controller decrements *Cerr* and proceeds to the next queue head in the asynchronous schedule.

65.4.3.12.1.2 Asynchronous - Do Complete Split

This state is entered from the Do Start Split state only after a start-split transaction receives an Ack handshake from the transaction translator.

For queue heads in this state, the host controller executes a complete-split transaction to the appropriate transaction translator. If the transaction translator responds with a Nyet handshake, the queue head is left in this state, the error counter is reset and the host controller proceeds to the next queue head in the asynchronous schedule. When a Nyet handshake is received for a bus transaction where the queue head's *PidCode* indicates an IN or OUT, the host controller reloads the error counter (*CErr*). When a Nyet handshake is received for a complete-split bus transaction where the queue head's *PidCode* indicates a SETUP, the host controller must not adjust the value of *CErr*.

Independent of *PIDCode*, the following responses have the effects:

- Transaction Error (XactErr). Timeout or data CRC failure, and so on. The error counter (*Cerr*) is decremented by one and the complete split transaction is *immediately* retried (if possible). If there is not enough time in the micro-frame to execute the retry, the host controller MUST ensure that the next time the host controller begins executing from the Asynchronous schedule, it must begin executing from this queue head. If another start-split (for some other endpoint) is sent to the transaction translator before the complete-split is really completed, the transaction translator could dump the results (which were never delivered to the host). This is why the core specification states the retries must be immediate. A method to

accomplish this behavior is to not advance the asynchronous schedule. When the host controller returns to the asynchronous schedule in the next micro-frame, the first transaction from the schedule is the retry for this endpoint.

If *Cerr* went to zero, the host controller must halt the queue.

- NAK. The target endpoint Nak'd the full- or low-speed transaction. The state of the transfer is not advanced and the state is exited. If the *PidCode* is a SETUP, then the Nak response is a protocol error. The *XactErr* status bit is set to one and the *CErr* field is decremented.
- STALL. The target endpoint responded with a STALL handshake. The host controller sets the *halt* bit in the status byte, retires the qTD but does not attempt to advance the queue.

If the *PidCode* indicates an IN, then any of following responses are expected:

- DATA0/1. On reception of data, the host controller ensures the PID matches the expected data toggle and checks CRC. If the packet is *good*, the host controller advances the state of the transfer, for example move the data pointer by the number of bytes received, decrement *BytesToTransfer* field by the number of bytes received, and toggle the *dt* bit. The host controller then exit this state. The response and advancement of transfer may trigger other processing events, such as retirement of the qTD and advancement of the queue.

If the data sequence PID does not match the expected, the data is ignored, the transfer state is not advanced and this state is exited. If the *PidCode* indicates an OUT/SETUP, then any of following responses are expected:

- ACK. The target endpoint accepted the data, so the host controller must advance the state of the transfer. The *Current Offset* field is incremented by *Maximum Packet Length* or *Bytes to Transfer*, whichever is less. The field *Bytes To Transfer* is decremented by the same amount and the data toggle bit (*dt*) is toggled. The host controller then exit this state.
- Advancing the transfer state may cause other processing events such as retirement of the qTD and advancement of the queue (see Managing Control/Bulk/Interrupt Transfers through Queue Heads).

65.4.3.12.2 Split Transaction Interrupt

Split-transaction Interrupt-IN/OUT endpoints are managed through the same data structures used for high-speed interrupt endpoints. They both co-exist in the periodic schedule.

Queue heads/qTDs offer the set of features required for reliable data delivery, which is characteristic to interrupt transfer types. The split-transaction protocol is managed completely within this defined functional transfer framework. For example, for a high-speed endpoint, the host controller visits a queue head, execute a high-speed transaction (if criteria are met) and advance the transfer state (or not) depending on the results of the entire transaction. For low- and full-speed endpoints, the details of the *execution* phase are different (that is takes more than one bus transaction to complete), but the remainder of the operational framework is intact. This means that the transfer advancement, and so on, occurs as defined in Managing Control/Bulk/Interrupt Transfers through Queue Heads, but only occurs on the completion of a split transaction.

65.4.3.12.2.1 Split Transaction Scheduling Mechanisms for Interrupt

Full- and low-speed Interrupt queue heads have an *EPS* field indicating full- or low-speed and have a non-zero *S-mask* field.

The host controller can detect this combination of parameters and assume the endpoint is a periodic endpoint. Low- and full-speed interrupt queue heads require the use of the split transaction protocol. The host controller sets the Endpoint Type (ET) field in the split token to indicate the transaction is an interrupt. These transactions are managed through a transaction translator's periodic pipeline. Software should not set these fields to indicate the queue head is an interrupt unless the queue head is used in the periodic schedule.

System software manages the per/transaction translator periodic pipeline by budgeting and scheduling exactly during which micro-frames the start-splits and complete-splits for each endpoint occurs. The characteristics of the transaction translator are such that the high-speed transaction protocol must execute during explicit micro-frames, or the data or response information in the pipeline is lost.

The following figure illustrates the general scheduling boundary conditions that are supported by the EHCI periodic schedule and queue head data structure. The S and ^{C}X labels indicate micro-frames where software can schedule start-splits and complete splits (respectively).

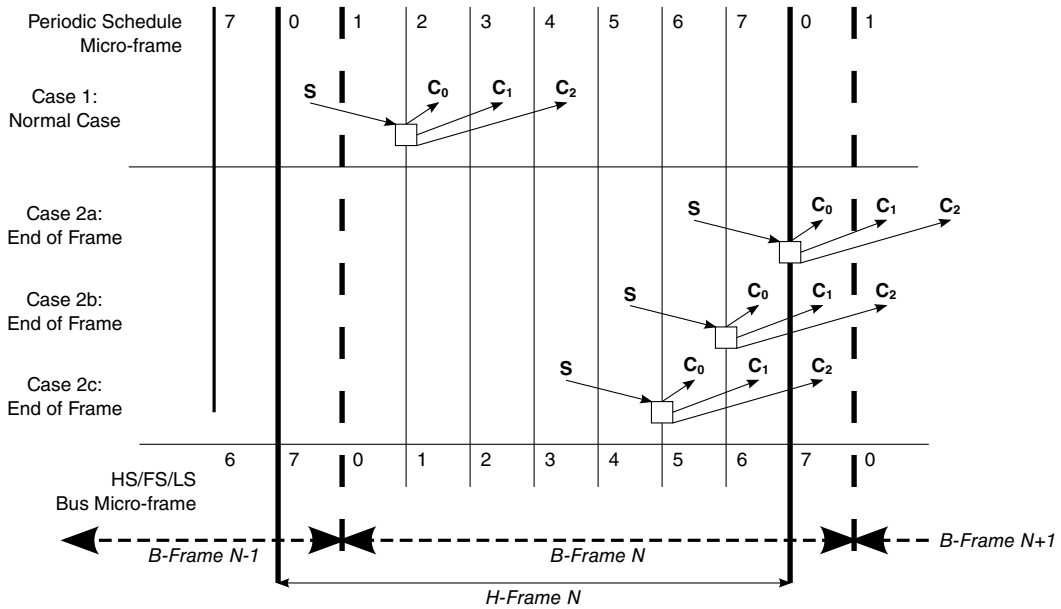


Figure 65-21. Split Transaction, Interrupt Scheduling Boundary Conditions

The scheduling cases are:

- Case 1: The normal scheduling case is where the entire split transaction is completely bounded by a frame (*H-Frame* in this case).
- Case 2a through Case 2c: The USB 2.0 Hub pipeline rules states clearly, when and how many complete-splits must be scheduled to account for earliest to latest execution on the full/low-speed link. The complete-splits may span the *H-Frame* boundary when the start-split is in micro-frame 4 or later. When this occurs, the *H-Frame* to *B-Frame* alignment requires that the queue head be reachable from consecutive periodic frame list locations. System software cannot build an efficient schedule that satisfies this requirement unless it uses FSTNs.

The figure below illustrates the general layout of the periodic schedule.

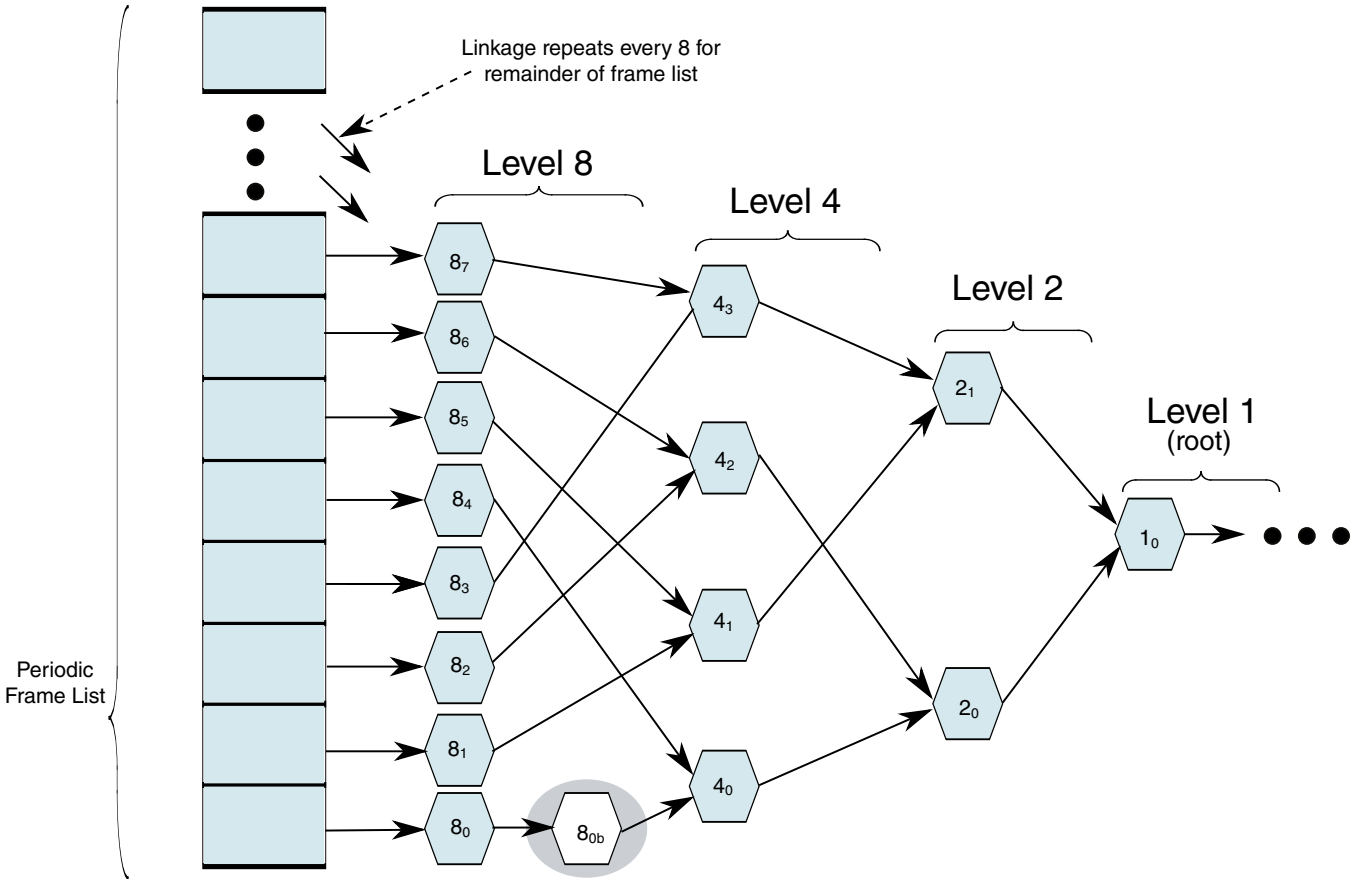


Figure 65-22. General Structure of EHCI Periodic Schedule Utilizing Interrupt Spreading

The periodic frame list is effectively the leaf level a binary tree, which is always traversed leaf to root. Each level in the tree corresponds to a 2^N poll rate. Software can efficiently manage periodic bandwidth on the USB by *spreading* interrupt queue heads that have the same poll rate requirement across all the available paths from the frame list. For example, system software can schedule eight poll rate 8 queue heads and account for them once in the high-speed bus bandwidth allocation.

When an endpoint is allocated an execution footprint that spans a frame boundary, the queue head for the endpoint must be reachable from consecutive locations in the frame list. An example would be if 8_{0b} where such an endpoint. Without additional support on the interface, to get 8_{0b} reachable at the correct time, software would have to link 8_1 to 8_{0b}. It would then have to move 4_1 and everything linked after into the same path as 4_0. This upsets the integrity of the binary tree and disallows the use of the spreading technique.

FSTN data structures are used to preserve the integrity of the binary-tree structure and enable the use of the spreading technique. Host Controller Operational Model for FSTNs defines the hardware and software operational model requirements for using FSTNs.

The following queue head fields are initialized by system software to instruct the host controller when to execute portions of the split-transaction protocol:

- *SplitXState*. This is single bit residing in the *Status* field of a queue head (see Table 65-23). This bit is used to track the current state of the split transaction.
- *Frame S-mask*. This is a bit-field where-in system software sets a bit corresponding to the micro-frame (within an *H-Frame*) that the host controller should execute a start-split transaction. This is always qualified by the value of the *SplitXState* bit in the *Status* field of the queue head. For example, referring to Figure 65-21, case one, the *S-mask* would have a value of 00000001b indicating that if the queue head is traversed by the host controller, and the *SplitXState* indicates Do_Start, and the current micro-frame as indicated by FRINDEX[2:0] is 0, then execute a start-split transaction.
- *Frame C-mask.* This is a bit-field where system software sets one or more bits corresponding to the micro-frames (within an *H-Frame*) that the host controller should execute complete-split transactions. The interpretation of this field is always qualified by the value of the *SplitXState* bit in the *Status* field of the queue head. For example, referring to Figure 65-21, case one, the *C-mask* would have a value of 00011100b indicating that if the queue head is traversed by the host controller, and the *SplitXState* indicates Do_Complete, and the current micro-frame as indicated by FRINDEX[2:0] is 2, 3, or 4, then execute a complete-split transaction. It is software's responsibility to ensure that the translation between *H-Frames* and *B-Frames* is correctly performed when setting bits in *S-mask* and *C-mask*

65.4.3.12.2.2 Host Controller Operational Model for FSTNs

The FSTN data structure is used to manage Low/Full-speed interrupt queue heads that need to be reached from consecutive frame list locations (that is boundary cases 2a through 2c).

An FSTN is essentially a *back pointer*, similar in intent to the back pointer field in the siTD data structure (see siTD Back Link Pointer).

This feature provides software a simple primitive to save a schedule position, redirect the host controller to traverse the necessary queue heads in the previous frame, then restore the original schedule position and complete normal traversal.

The four components to the use of FSTNs:

- FSTN data structure.
- A *Save Place* indicator. This is always an FSTN with its *Back Path Link Pointer.T-bit* set to zero.

- A *Restore* indicator. This is always an FSTN with its *Back Path Link Pointer.T-bit* set to one.
- Host controller FSTN traversal rules.

When the host controller encounters an FSTN during micro-frames 2 through 7 it simply follows the node's *Normal Path Link Pointer* to access the next schedule data structure.

NOTE

The FSTN's *Normal Path Link Pointer.T-bit* may set to one, which the host controller must interpret as the end of periodic list mark.

When the host controller encounters a *Save-Place* FSTN in micro-frames 0 or 1, it saves the value of the *Normal Path Link Pointer* and set an internal flag indicating that it is executing in *Recovery Path* mode. *Recovery Path* mode modifies the host controller's rules for how it traverses the schedule and limits which data structures is considered for execution of bus transactions. The host controller continues executing in *Recovery Path* mode until it encounters a *Restore* FSTN or it determines that it has reached the end of the micro-frame (see details in the list below).

The rules for schedule traversal and limited execution while in *Recovery Path* mode are:

- Always follow the *Normal Path Link Pointer* when it encounters an FSTN that is a *Save-Place* indicator. The host controller must not recursively follow *Save-Place* FSTNs. Therefore, while executing in *Recovery Path* mode, it must never follow an FSTN's *Back Path Link Pointer*.
- Do not process an siTD or, iTD data structure. Simply follow its *Next Link Pointer*.
- Do not process a QH (Queue Head) whose *EPS* field indicates a high-speed device. Simply follow its *Horizontal Link Pointer*.
- When a QH's *EPS* field indicates a Full/Low-speed device, the host controller considers only it for execution if its *SplitXState* is DoComplete (note: this applies whether the *PID Code* indicates an IN or an OUT). See Execute Transaction and Tracking Split Transaction Progress for Interrupt Transfers for a complete list of additional conditions that must be met in general for the host controller to issue a bus transaction.
  - The host controller must not execute a Start-split transaction while executing in *Recovery Path* mode. See Periodic Isochronous - Do Complete Split for special handling when in *Recovery Path* mode.
- Stop traversing the *recovery path* when it encounters an FSTN that is a *Restore* indicator. The host controller unconditionally uses the saved value of the *Save-Place* FSTN's *Normal Path Link Pointer* when returning to the normal path traversal. The

host controller must clear the context of executing a *Recovery Path* when it restores schedule traversal to the *Save-Place* FSTN's *Normal Path Link Pointer*.

- If the host controller determines that there is not enough time left in the micro-frame to complete processing of the periodic schedule, it abandons traversal of the recovery path, and clears the context of executing a recovery path. The result is that at the start of the next consecutive micro-frame, the host controller starts traversal at the frame list.

An example traversal of a periodic schedule that includes FSTNs is illustrated in the following figure.

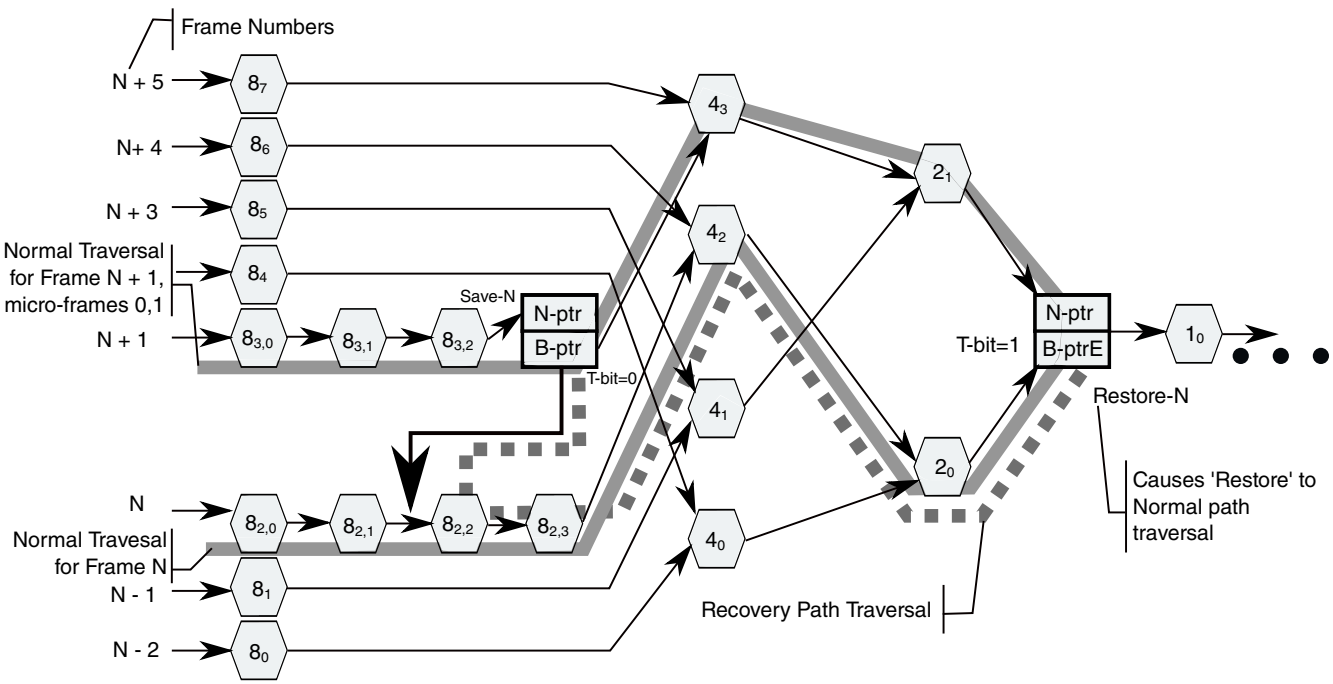


Figure 65-23. Example Host Controller Traversal of Recovery Path via FSTNs

In frame N+1 (micro-frames 0 and 1), when the host controller encounters Save-Path FSTN (Save-N), it observes that Save-N.Back Path Link Pointer.T-bit is zero (definition of a Save-Path indicator). The host controller saves the value of Save-N.Normal Path Link Pointer and follows Save-N.Back Path Link Pointer. At the same time, it sets an internal flag indicating that it is now in *Recovery Path* mode (the recovery path is annotated in the figure above with a large dashed line). The host controller continues traversing data structures on the recovery path and executing only those bus transactions as noted above, on the recovery path until it reaches Restore FSTN (Restore-N). Restore-N.Back Path Link Pointer.T-bit is set to one (definition of a Restore indicator), so the host controller exits *Recovery Path* mode by clearing the internal *Recovery Path* mode flag and commences (restores) schedule traversal using the saved value of the *Save-Place* FSTN's *Normal Path Link Pointer* (for example Save-N.Normal Path Link Pointer). The nodes traversed during these micro-frames include: {$8_{3.0}$, $8_{3.1}$, $8_{3.2}$, Save-A, $8_{2.2}$, $8_{2.3}$, 4_2,

2_0, Restore-N, 4_3, 2_1, Restore-N, 1_0 …}. The nodes on the recovery-path are in bold. In frame N (micro-frames 0-7), for this example, the host controller traverses all of the schedule data structures utilizing the *Normal Path Link Pointers* in any FSTNs it encounters. This is because the host controller has not yet encountered a *Save-Place* FSTN so it not executing in *Recovery Path* mode. When it encounters the *Restore* FSTN, (Restore-N), during micro-frames 0 and 1, it uses Restore-N.Normal Path Link Pointer to traverse to the next data structure (that is normal schedule traversal). This is because the host controller must use a Restore FSTN's *Normal Path Link Pointer* when not executing in a *Recovery-Path* mode. The nodes traversed during frame N include: {$8_{2.0}$, $8_{2.1}$, $8_{2.2}$, $8_{2.3}$, 4_2, 2_0, Restore-N, 1_0 …}.

In frame N+1 (micro-frames 2-7), when the host controller encounters Save-Path FSTN Save-N, it unconditionally follows Save-N.Normal Path Link Pointer. The nodes traversed during these micro-frames include: {$8_{3.0}$, $8_{3.1}$, $8_{3.2}$, Save-A, 4_3, 2_1, Restore-N, 1_0 …}.

65.4.3.12.2.3 Software Operational Model for FSTNs

Software must create a consistent, coherent schedule for the host controller to traverse.

When using FSTNs, system software must adhere to the following rules:

- Each *Save-Place* indicator requires a matching *Restore* indicator.
  - The *Save-Place* indicator is an FSTN with a valid *Back Path Link Pointer and T-bit* equal to zero.
    - *Back Path Link Pointer.Typ* field must be set to indicate the referenced data structure is a queue head. The *Restore* indicator is an FSTN with its *Back Path Link Pointer.T-bit* set to one.
  - A *Restore* FSTN may be matched to one or more *Save-Place* FSTNs. For example, if the schedule includes a poll-rate 1 level, then system software only needs to place a *Restore* FSTN at the beginning of this list in order to match all possible *Save-Place* FSTNs.
- If the schedule does not have elements linked at a poll-rate level of one, and one or more *Save-Place* FSTNs are used, then System Software must ensure the *Restore* FSTN's *Normal Path Link Pointer*'s *T-bit* is set to one, as this is used to mark the end of the periodic list.
- When the schedule does have elements linked at a poll rate level of one, a *Restore* FSTN must be the first data structure on the poll rate one list. All traversal paths from the frame list converge on the poll-rate one list. System software must ensure that *Recovery Path* mode is exited before the host controller is allowed to traverse the poll rate level one list.
- A *Save-Place* FSTN's *Back Path Link Pointer* must reference a queue head data structure. The referenced queue head must be reachable from the previous frame list

location. In other words, if the *Save-Place* FSTN is reachable from frame list offset N, then the FSTN's *Back Path Link Pointer* must reference a queue head that is reachable from frame list offset N-1.

Software should make the schedule as efficient as possible. What this means in this context is that software should have no more than one *Save-Place* FSTN reachable in any single frame. Note there is times when two (or more, depending on the implementation) could exist as full/low-speed footprints change with bandwidth adjustments. This could occur, for example when a bandwidth re-balance causes system software to move the *Save-Place* FSTN from one poll rate level to another. During the transition, software must preserve the integrity of the previous schedule until the new schedule is in place.

65.4.3.12.2.4 Tracking Split Transaction Progress for Interrupt Transfers

To correctly maintain the data stream, the host controller must be able to detect and report errors where data is lost.

For interrupt-IN transfers, data is lost when it makes it into the USB 2.0 hub, but the USB 2.0 host system is unable to get it from the USB 2.0 Hub and into the system before it expires from the transaction translator pipeline.

When a lost data condition is detected, the queue must be halted, thus signaling system software to recover from the error. A data-loss condition exists whenever a start-split is issued, accepted and successfully executed by the USB 2.0 Hub, but the complete-splits get unrecoverable errors on the high-speed link, or the complete-splits do not occur at the correct times. One reason complete-splits might not occur at the right time would be due to host-induced system hold-offs that cause the host controller to miss bus transactions because it cannot get timely access to the schedule in system memory.

The same condition can occur for an interrupt-OUT, but the result is not an endpoint halt condition, but rather effects only the progress of the transfer. The queue head has the following fields to track the progress of each split transaction. These fields are used to keep incremental state about which (and when) portions have been executed.

- *C-prog-mask*. This is an eight-bit bit-vector where the host controller keeps track of which complete-splits have been executed. Due to the nature of the Transaction Translator periodic pipeline, the complete-splits need to be executed in-order. The host controller needs to detect when the complete-splits have not been executed in order. This can only occur due to system hold-offs where the host controller cannot get to the memory-based schedule. *C-prog-mask* is a simple bit-vector that the host controller sets one of the *C-prog*-mask bits for each complete-split executed. The bit position is determined by the micro-frame number in which the complete-split was executed. The host controller always checks *C-prog-mask* before executing a

complete-split transaction. If the previous complete-splits have not been executed then it means one (or more) have been skipped and data has potentially been lost.
- *FrameTag*. This field is used by the host controller during the complete-split portion of the split transaction to tag the queue head with the frame number (*H-Frame* number) when the next complete split must be executed.
- *S-bytes*. This field can be used to store the number of data payload bytes sent during the start-split (if the transaction was an OUT). The *S-bytes* field must be used to accumulate the data payload bytes received during the complete-splits (for an IN).

65.4.3.12.2.5 Split Transaction Execution State Machine for Interrupt

In the following presentation, all references to micro-frame are in the context of a micro-frame within an *H-Frame.*

>As with asynchronous Full- and Low-speed endpoints, a split-transaction state machine is used to manage the split transaction sequence.

Aside from the fields defined in the queue head for scheduling and tracking the split transaction, the host controller calculates one internal mechanism that is also used to manage the split transaction. The internal calculated mechanism is:

- *cMicroFrameBit*. This is a single-bit encoding of the current micro-frame number. It is an eight-bit value calculated by the host controller at the beginning of every micro-frame. It is calculated from the three least significant bits of the *FRINDEX* register (that is, *cMicroFrameBit* = (1 shifted-left(*FRINDEX*[2:0])). The *cMicroFrameBit* has at most one bit asserted, which always corresponds to the current micro-frame number. For example, if the current micro-frame is 0, then *cMicroFrameBit* will equal 00000001b. The variable *cMicroFrameBit* is used to compare against the *S-mask* and *C-mask* fields to determine whether the queue head is marked for a start- or complete-splt transaction for the current micro-frame.

The following figure illustrates the state machine for managing a complete interrupt split transaction. There are two phases to each split transaction. The first is a single start-split transaction, which occurs when the *SplitXState* is at Do_Start and the single bit in *cMicroFrameBit* has a corresponding bit active in *QH.S-mask*. The transaction translator does not acknowledge the receipt of the periodic start-split, so the host controller unconditionally transitions the state to Do_Complete. Due to the available jitter in the transaction translator pipeline, there will be more than one complete-split transaction scheduled by software for the Do_Complete state. This translates simply to the fact that there are multiple bits set to a one in the *QH.C-mask* field.

The host controller keeps the queue head in the Do_Complete state until the split transaction is complete (see definition below), or an error condition triggers the *three-strikes-rule* (for example, after the host tries the same transaction three times, and each

encounters an error, the host controller will stop retrying the bus transaction and halt the endpoint, thus requiring system software to detect the condition and perform system-dependent recovery).

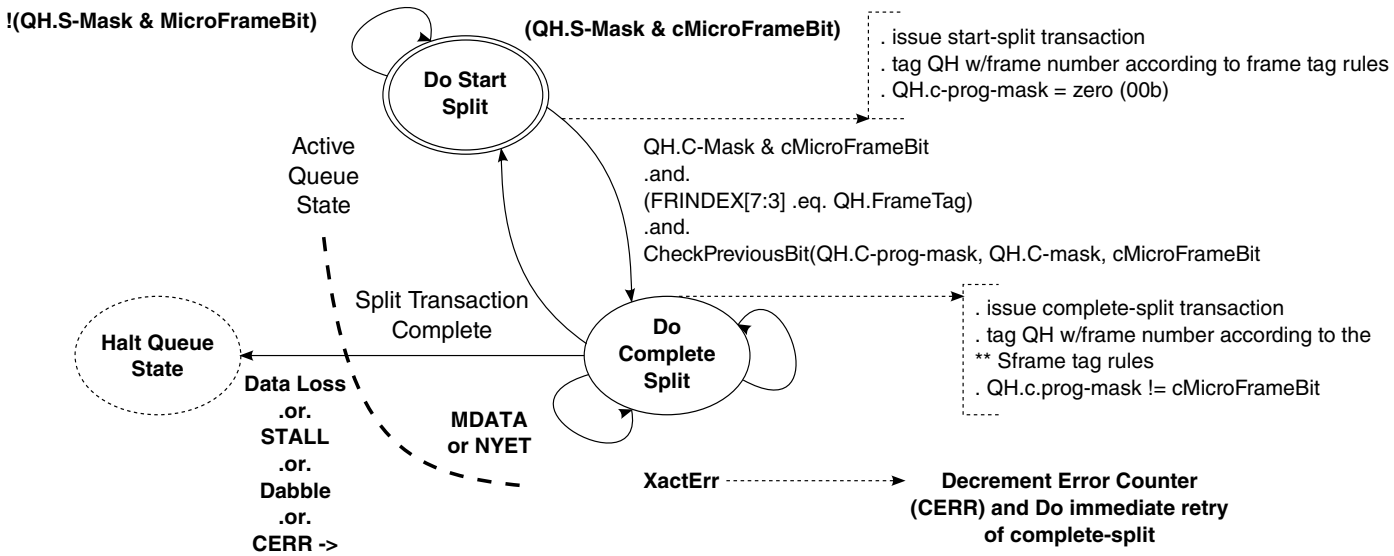


Figure 65-24. Split Transaction State Machine for Interrupt

See Previous Section for the frame tag management rules.

Periodic Interrupt - Do Start Split

This is the state software must initialize a full- or low-speed interrupt queue head *StartXState* bit. This state is entered from the Do_Complete Split state only after the split transaction is complete. This occurs when one of the following events occur: The transaction translator responds to a complete-split transaction with one of the following:

- NAK. A NAK response is a propagation of the full- or low-speed endpoint's NAK response.
- ACK. An ACK response is a propagation of the full- or low-speed endpoint's ACK response. Only occurs on an OUT endpoint.
- DATA 0/1. Only occurs for INs. Indicates that this is the last of the data from the endpoint for this split transaction.
- ERR. The transaction on the low-/full-speed link below the transaction translator had a failure (for example, timeout, bad CRC, etc.).
- NYET (and Last). The host controller issued the last complete-split and the transaction translator responded with a NYET handshake. This means that the start-split was not correctly received by the transaction translator, so it never executed a transaction to the full- or low-speed endpoint, see Section Periodic Isochronous - Do Complete Split for the definition of 'Last'.

Each time the host controller visits a queue head in this state (once within the Execute Transaction state), it performs the following test to determine whether to execute a start-split.

- *QH.S-mask* is bit-wise anded with *cMicroFrameBit*.

If the result is non-zero, then the host controller will issue a start-split transaction. If the *PIDCode* field indicates an IN transaction, the host controller must zero-out the *QH.S-bytes* field. After the split-transaction has been executed, the host controller sets up state in the queue head to track the progress of the complete-split phase of the split transaction. Specifically, it records the expected frame number into *QH.FrameTag* field (see Section), set *C-prog-mask* to zero (00h), and exits this state. Note that the host controller must not adjust the value of *CErr* as a result of completion of a start-split transaction.

Periodic Interrupt - Do Complete Split

This state is entered unconditionally from the Do Start Split state after a start-split transaction is executed on the bus. Each time the host controller visits a queue head in this state (once within the Execute Transaction state), it checks to determine whether a complete-split transaction should be executed now.

There are four tests to determine whether a complete-split transaction should be executed.

- Test A. *cMicroFrameBit* is bit-wise anded with *QH.C-mask* field. A non-zero result indicates that software scheduled a complete-split for this endpoint, during this micro-frame.
- Test B. *QH.FrameTag* is compared with the current contents of *FRINDEX*[7:3]. An equal indicates a match.
- Test C. The complete-split progress bit vector is checked to determine whether the previous bit is set, indicating that the previous complete-split was appropriately executed. An example algorithm for this test is provided below:

```
Algorithm Boolean CheckPreviousBit(QH.C-prog-mask, QH.C-mask, cMicroFrameBit)
Begin
-- Return values:
-- TRUE - no error
-- FALSE - error
--
Boolean rvalue = TRUE;
previousBit = cMicroframeBit logical-rotate-right(1)
-- Bit-wise anding previousBit with C-mask indicates
-- whether there was an intent
-- to send a complete split in the previous micro-frame. So,
-- if the
-- 'previous bit' is set in C-mask, check C-prog-mask to
-- make sure it
-- happened.
If (previousBit bitAND QH.C-mask)then
                                If not(previousBit bitAND QH.C-prog-mask) then
                      rvalue = FALSE;
          End if
End If
-- If the C-prog-mask already has a one in this bit position,
```

```
-- then an aliasing
-- error has occurred. It will probably get caught by the
-- FrameTag Test, but
-- at any rate it is an error condition that as detectable here
-- should not allow
-- a transaction to be executed.
If (cMicroFrameBit bitAND QH.C-prog-mask) then
 rvalue = FALSE;
End if
return (rvalue)
End Algorithm
```

- Test D. Check to see if a start-split should be executed in this micro-frame. Note this is the same test performed in the Do Start Split state (see Section Periodic Isochronous - Do Start Split). Whenever it evaluates to TRUE and the controller is NOT processing in the context of a *Recovery Path* mode, it means a start-split should occur in this micro-frame. Test D and Test A evaluating to TRUE at the same time is a system software error. Behavior is undefined.

If (A .and. B .and. C .and. not(D)) then the host controller will execute a complete-split transaction. When the host controller commits to executing the complete-split transaction, it updates *QH.C-prog-mask* by bit-ORing with *cMicroFrameBit*. On completion of the complete-split transaction, the host controller records the result of the transaction in the queue head and sets *QH.FrameTag* to the expected *H-Frame* number (see Section). The effect to the state of the queue head and thus the state of the transfer depends on the response by the transaction translator to the complete-split transaction. The following responses have the effects (note that any responses that result in decrementing of the *CErr* will result in the queue head being halted by the host controller if the result of the decrement is zero):

- NYET (and Last). On each NYET response, the host controller checks to determine whether this is the last complete-split for this split transaction. Last is defined in this context as the condition where all of the scheduled complete-splits have been executed. If it is the last complete-split (with a NYET response), then the transfer state of the queue head is not advanced (never received any data) and this state exited. The transaction translator must have responded to all the clompete-splits with NYETs, meaning that the start-split issued by the host controller was not received. The start-split should be retried at the next poll period.
- The test for whether this is the Last complete split can be performed by XOR *QH.C-mask* with *QH.C-prog-mask*. If the result is all zeros then all complete-splits have been executed. When this condition occurs, the *XactErr* status bit is set to a one and the *CErr* field is decremented.
- NYET (and not Last). See above description for testing for Last. The complete-split transaction received a NYET response from the transaction translator. Do not update any transfer state (except for *C-prog-mask* and *FrameTag*) and stay in this state. The host controller must not adjust *CErr* on this response.

- Transaction Error (XactErr). Timeout, data CRC failure, etc. The *CErr* field is decremented and the *XactErr* bit in the *Status* field is set to a one. The complete split transaction is *immediately* retried (if *Cerr* is non-zero).If there is not enough time in the micro-frame to complete the retry and the endpoint is an IN, or *CErr* is decremented to a zero from a one, the queue is halted. If there is not enough time in the micro-frame to complete the retry and the endpoint is an OUT and *CErr* is not zero, then this state is exited (that is, return to Do Start Split). This results in a retry of the entire OUT split transaction, at the next poll period. Refer to Chapter 11 Hubs (specifically the section full- and low-speed Interrupts) in the USB Specification Revision 2.0 for detailed requirements on why these errors must be immediately retried.
- ACK. This can only occur if the target endpoint is an OUT. The target endpoint ACK'd the data and this response is a propagation of the endpoint ACK up to the host controller. The host controller must advance the state of the transfer. The *Current Offset* field is incremented by *Maximum Packet Length* or *Bytes to Transfer*, whichever is less. The field *Bytes To Transfer* is decremented by the same amount. And the data toggle bit (*dt*) is toggled. The host controller will then exit this state for this queue head. The host controller must reload *CErr* with maximum value on this response. Advancing the transfer state may cause other process events such as retirement of the qTD and advancement of the queue (see Section Managing Control/Bulk/Interrupt Transfers through Queue Heads).
- MDATA. This response will only occur for an IN endpoint. The transaction translator responded with zero or more bytes of data and an MDATA PID. The incremental number of bytes received is accumulated in *QH.S-bytes*. The host controller must not adjust *CErr* on this response.
- DATA0/1. This response may only occur for an IN endpoint. The number of bytes received is added to the accumulated byte count in *QH.S-bytes*. The state of the transfer is advanced by the result and the host controller will exit this state for this queue head.
- Advancing the transfer state may cause other processing events such as retirement of the qTD and advancement of the queue (see Section Managing Control/Bulk/Interrupt Transfers through Queue Heads).
- If the data sequence PID does not match the expected, the entirety of the data received in this split transaction is ignored, the transfer state is not advanced and this state is exited.
- NAK. The target endpoint Nak'd the full- or low-speed transaction. The state of the transfer is not advanced, and this state is exited. The host controller must reload *CErr* with maximum value on this response.

- ERR. There was an error during the full- or low-speed transaction. The ERR status bit is set to a one, *Cerr* is decremented, the state of the transfer is not advanced, and this state is exited.
- STALL. The queue is halted (an exit condition of the Execute Transaction state). The status field bits: *Active* bit is set to zero and the *Halted* bit is set to a one and the qTD is retired. Responses which are not enumerated in the list or which are received out of sequence are illegal and may result in undefined host controller behavior. The other possible combinations of tests A, B, C, and D may indicate that data or response was lost. The table below lists the possible combinations and the appropriate action.

Table 65-47. Interrupt IN/OUT Do Complete Split State Execution Criteria

Condition	Action	Description
not(A) not(D)	Ignore QHD	Neither a start nor complete-split is scheduled for the current micro-frame.Host controller should continue walking the schedule.
A not(C)	If PIDCode = IN Halt QHD If PIDCode = OUT Retry start-split	Progress bit check failed. These means a complete-split has been missed. There is the possibility of lost data. If *PIDCode* is an IN, then the Queue head must be halted. If *PIDCode* is an OUT, then the transfer state is not advanced and the state exited (for example, start-split is retried). This is a host-induced error and does not effect *CER*R. In either case, set the *Missed Micro-frame* bit in the status field to a one.
A not(B) C	If PIDCode = IN Halt QHD If PIDCode = OUT Retry start-split	*QH.FrameTag* test failed. This means that exactly one or more *H-Frames* have been skipped. This means complete-splits and have missed. There is the possibility of lost data. If *PIDCode* is an IN, then the Queue head must be halted. If *PIDCode* is an OUT, then the transfer state is not advanced and the state exited (for example, start-split is retried). This is a host-induced error and does not effect *CER*R. In either case, set the *Missed Micro-frame* bit in the status field to a one.
A B C not(D)	Execute complete-split	This is the non-error case where the host controller executes a complete-split transaction.
D	If PIDCode = IN Halt QHD If PIDCode = OUT Retry start-split	This is a degenerate case where the start-split was issued, but all of the complete-splits were skipped and all possible intervening opportunities to detect the missed data failed to fire. If *PIDCode* is an IN, then the Queue head must be halted. If *PIDCode* is an OUT, then the transfer state is not advanced and the state exited (for example, start-split is retried). This is a host-induced error and does not effect *CER*R. In either case, set the *Missed Micro-frame* bit in the status field to a one. Note: When executing in the context of a *Recovery Path* mode, the host controller is allowed to process the queue head and take the actions indicated above, or it may wait until the queue head is visited in the

Table 65-47. Interrupt IN/OUT Do Complete Split State Execution Criteria

		normal processing mode. Regardless, the host controller must not execute a start-split in the context of a executing in a *Recovery Path* mode.

Managing QH.FrameTag Field

The *QH.FrameTag* field in a queue head is completely managed by the host controller. The rules for setting *QH.FrameTag* are simple:

- Rule 1: If transitioning from Do Start Split to Do Complete Split and the current value of *FRINDEX*[2:0] is 6 *QH.FrameTag* is set to *FRINDEX*[7:3] + 1. This accommodates split transactions whose start-split and complete-splits are in different *H-Frames* (case 2a, see Figure 65-21).
- Rule 2: If the current value of *FRINDEX*[2:0] is 7, *QH.FrameTag* is set to *FRINDEX*[7:3] + 1. This accommodates staying in Do Complete Split for cases 2a, 2b, and 2c (Figure 65-21).
- Rule 3: If transitioning from Do_Start Split to Do Complete Split and the current value of *FRINDEX*[2:0] is not 6, or currently in Do Complete Split and the current value of (*FRINDEX*[2:0]) is not 7, *FrameTag* is set to *FRINDEX*[7:3]. This accommodates all other cases (Figure 65-21).

65.4.3.12.2.6 Rebalancing the periodic schedule

System software must occasionally adjust a periodic queue head's S-mask and C-mask fields during operation.

This need occurs when adjustments to the periodic schedule create a new bandwidth budget and one or more queue head's are assigned new execution footprints (that is, new S-mask and C-mask values).

It is imperative that System software must not update these masks to new values in the midst of a split transaction. In order to avoid any race conditions with the update, the EHCI host controller provides a simple assist to system software. System software sets the *Inactivate-on-next-Transaction* (*I*) bit to a one to signal the host controller that it intends to update the S-mask and C-mask on this queue head. System software will then wait for the host controller to observe the *I-bit* is a one and transition the *Active* bit to a zero. The rules for how and when the host controller sets the *Active* bit to zero are enumerated below:

- If the *Active* bit is a zero, no action is taken. The host controller does not attempt to advance the queue when the *I-bit* is a one.
- If the *Active* bit is a one and the *SplitXState* is DoStart (regardless of the value of *S-mask*), the host controller will simply set *Active* bit to a zero. The host controller is

not required to write the transfer state back to the *current* qTD. Note that if the *S-mask* indicates that a start-split is scheduled for the current micro-frame, the host controller must not issue the start-split bus transaction. It must set the *Active* bit to zero.

System software must save transfer state before setting the *I-bit* to a one. This is required so that it can correctly determine what transfer progress (if any) occurred after the *I-bit* was set to a one and the host controller executed its final bus-transaction and set *Active* to a zero.

After system software has updated the S-mask and C-mask, it must then reactivate the queue head. Because the *Active* bit and the *I-bit* cannot be updated with the same write, system software needs to use the following algorithm to coherently re-activate a queue head that has been stopped via the *I-bit*.

1. Set the *Halted* bit to a one, then
2. Set the *I-bit* to a zero, then
3. Set the *Active* bit to a one and the *Halted* bit to a zero in the same write.

Setting the *Halted* bit to a one inhibits the host controller from attempting to advance the queue between the time the *I-bit* goes to a zero and the *Active* bit goes to a one.

65.4.3.12.3 Split Transaction Isochronous

Full-speed isochronous transfers are managed using the split-transaction protocol through a USB 2.0 transaction translator in a USB2.0 Hub. The EHCI controller utilizes siTD data structure to support the special requirements of isochronous split-transactions.

This data structure uses the scheduling model of isochronous TDs (iTD, Section Isochronous (High-Speed) Transfer Descriptor (iTD)) (see Section Managing Isochronous Transfers Using iTDs for the operational model of iTDs) with the contiguous data feature provided by queue heads. This simple arrangement allows a single isochronous scheduling model and adds the additional feature that all data received from the endpoint (per split transaction) must land into a contiguous buffer.

65.4.3.12.3.1 Split Transaction Scheduling Mechanisms for Isochronous

Full-speed isochronous transactions are managed through a transaction translator's periodic pipeline. As with full- and low-speed interrupt, system software manages each transaction translator's periodic pipeline by budgeting and scheduling exactly during which micro-frames the start-splits and complete-splits for each full-speed isochronous endpoint occur.

The requirements described in Section Split Transaction Scheduling Mechanisms for Interrupt apply. The following figure illustrates the general scheduling boundary conditions that are supported by the EHCI periodic schedule. The ^{S}X and ^{C}X labels indicate micro-frames where software can schedule start- and complete-splits (respectively). The *H-Frame* boundaries are marked with a large, solid bold vertical line. The *B-Frame* boundaries are marked with a large, bold, dashed line. The bottom of the figure illustrates the relationship of an siTD to the *H-Frame*.

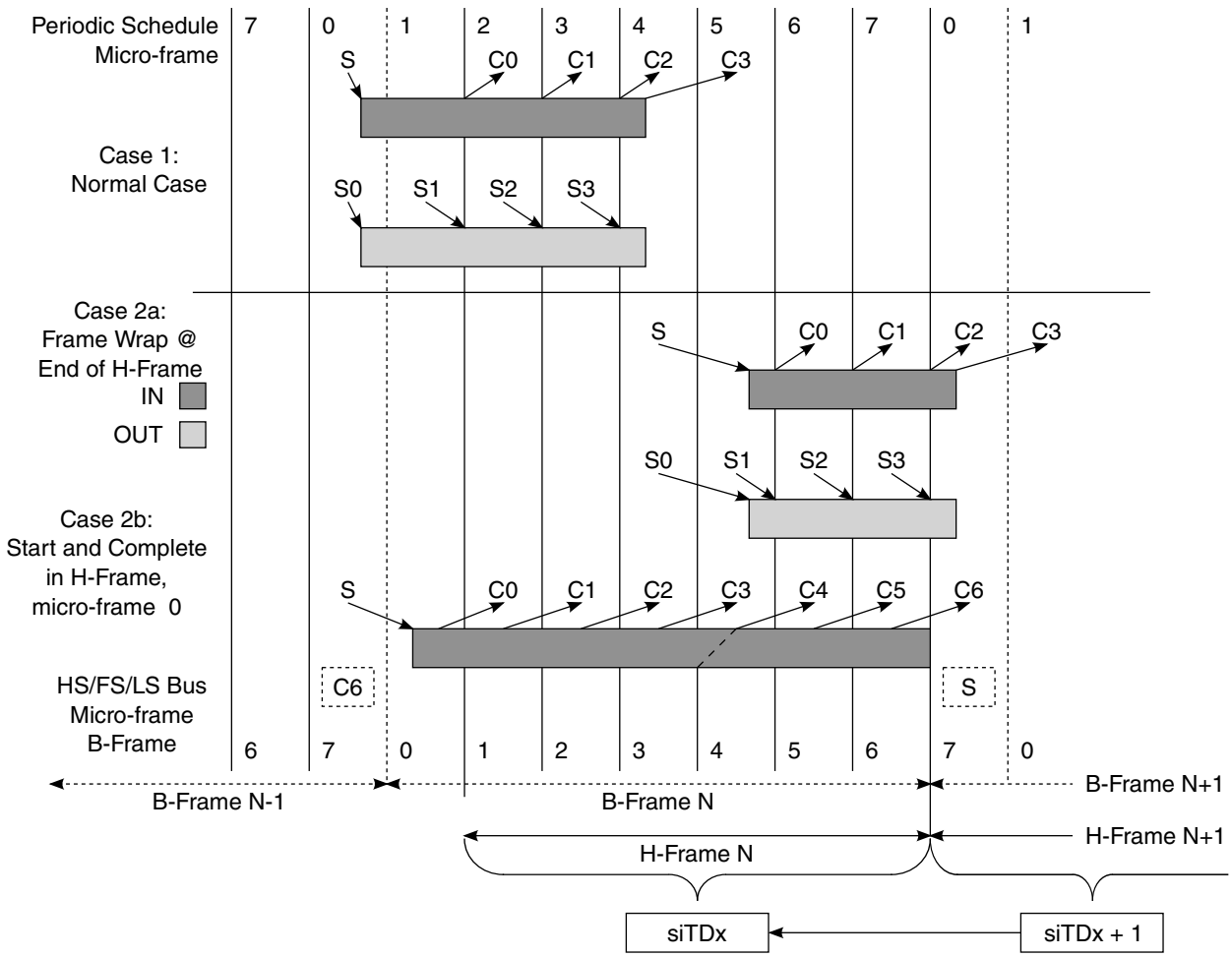


Figure 65-25. Split Transaction, Isochronous Scheduling Boundary Conditions

When the endpoint is an isochronous OUT, there are only start-splits, and no complete-splits. When the endpoint is an isochronous IN, there is at most one start-split and one to *N* complete-splits. The scheduling boundary cases are:

- *Case 1*: The entire split transaction is completely bounded by an *H-Frame*. For example: the start-splits and complete-splits are all scheduled to occur in the same *H-Frame*.

- *Case 2a*: This boundary case is where one or more (at most two) complete-splits of a split transaction IN are scheduled across an *H-Frame* boundary. This can only occur when the split transaction has the possibility of moving data in *B-Frame,* micro-frames 6 or 7 (*H-Frame* micro-frame 7 or 0). When an *H-Frame* boundary wrap condition occurs, the scheduling of the split transaction spans more than one location in the periodic list.(For example, it takes two siTDs in adjacent periodic frame list locations to fully describe the scheduling for the split transaction.)
- Although the scheduling of the split transaction may take two data structures, all of the complete-splits for each full-speed IN isochronous transaction must use only one data pointer. For this reason, siTDs contain a back pointer, the use of which is described below.
- Software must never schedule full-speed isochronous OUTs across an *H-Frame* boundary.
- *Case 2b*: This case can only occur for a very large isochronous IN. It is the only allowed scenario where a start-split and complete-split for the same endpoint can occur in the same micro-frame. Software must enforce this rule by scheduling the large transaction first. Large is defined to be anything larger than 579 byte maximum packet size.

A subset of the same mechanisms employed by full- and low-speed interrupt queue heads are employed in siTDs to schedule and track the portions of isochronous split transactions. The following fields are initialized by system software to instruct the host controller when to execute portions of the split transaction protocol.

- *SplitXState*. This is a single bit residing in the *Status* field of an siTD (see Figure 65-26). This bit is used to track the current state of the split transaction. The rules for managing this bit are described in Section Split Transaction Execution State Machine for Interrupt .
- *Frame S-mask*. This is a bit-field where-in system software sets a bit corresponding to the micro-frame (within an *H-Frame*) that the host controller should execute a start-split transaction. This is always qualified by the value of the *SplitXState* bit. For example, referring to the IN example in Figure 65-25, case one, the *S-mask* would have a value of 00000001b indicating that if the siTD is traversed by the host controller, and the *SplitXState* indicates Do Start Split, and the current micro-frame as indicated by USB_n_FRINDEX[2:0] is 0, then execute a start-split transaction.
- *Frame C-mask.* This is a bit-field where system software sets one or more bits corresponding to the micro-frames (within an *H-Frame*) that the host controller should execute complete-split transactions. The interpretation of this field is always qualified by the value of the *SplitXState* bit. For example, referring to the IN example in Figure 65-25, case one, the *C-mask* would have a value of 00111100b indicating that if the siTD is traversed by the host controller, and the *SplitXState* indicates Do

Complete Split, and the current micro-frame as indicated by *USB_n_FRINDEX*[2:0] is 2, 3, 4, or 5, then execute a complete-split transaction.

- *Back Pointer*. This field in a siTD is used to complete an IN split-transaction using the previous *H-Frame*'s siTD. This is only used when the scheduling of the complete-splits span an *H-Frame* boundary.

There exists a one-to-one relationship between a high-speed isochronous split transaction (including all start- and complete-splits) and one full-speed isochronous transaction. An siTD contains (amongst other things) buffer state and split transaction scheduling information. An siTD's buffer state always maps to one full-speed isochronous data payload. This means that for any full-speed transaction payload, a single siTD's data buffer must be used. This rule applies to both IN an OUTs. An siTD's scheduling information usually also maps to one high-speed isochronous split transaction. The exception to this rule is the *H-Frame* boundary wrap cases mentioned above.

The siTD data structure describes at most, one frame's worth of high-speed transactions and that description is strictly bounded within a frame boundary. The figure below illustrates some examples. On the top are examples of the full-speed transaction footprints for the boundary scheduling cases described above. In the middle are time-frame references for both the *B-Frames* (HS/FS/LS Bus) and the *H-Frames*. On the bottom is illustrated the relationship between the scope of an siTD description and the time references. Each *H-Frame* corresponds to a single location in the periodic frame list. The implication is that each siTD is reachable from a single periodic frame list location at a time.

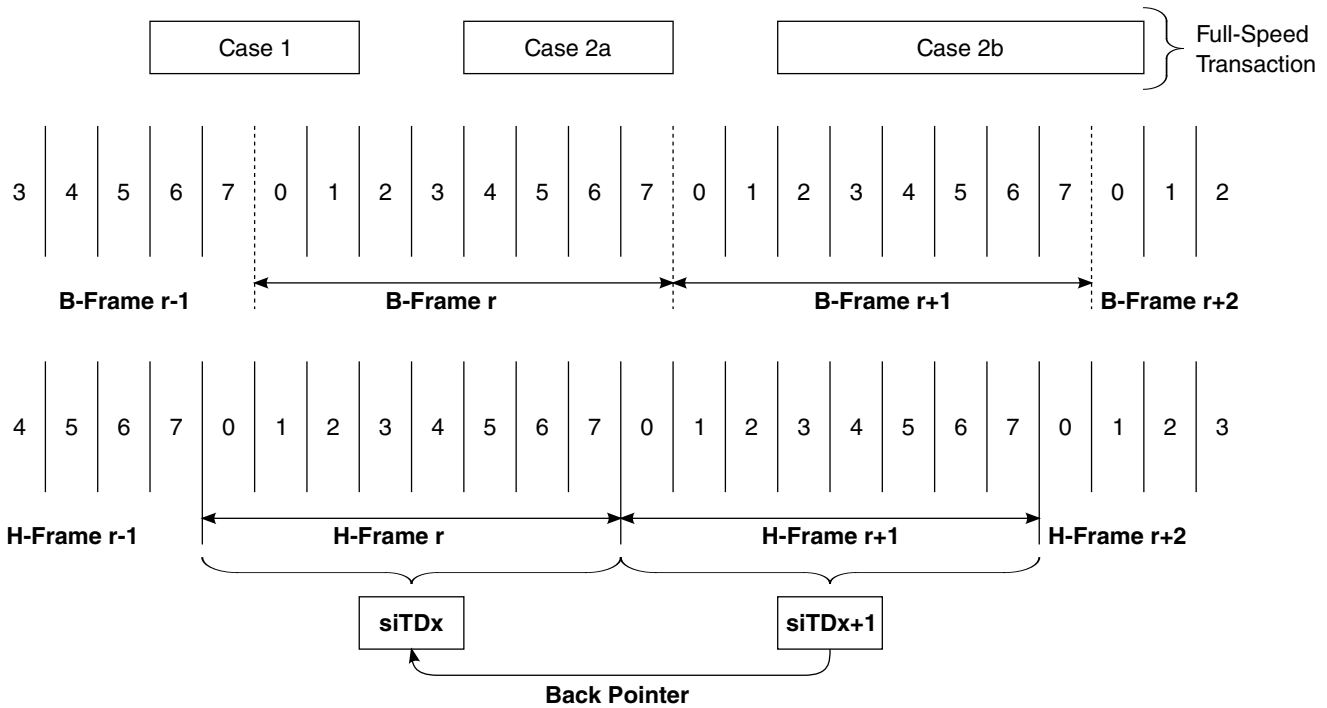


Figure 65-26. siTD Scheduling Boundary Examples

Each case is described below:

- *Case 1*: One siTD is sufficient to describe and complete the isochronous split transaction because the whole isochronous split transaction is tightly contained within a single *H-Frame*.
- *Case 2a, 2b*: Although both INs and OUTs can have these footprints, OUTs always take only one siTD to schedule. However, INs (for these boundary cases) require two siTDs to complete the scheduling of the isochronous split transaction $siTD_X$ is used to always issue the start-split and the first *N* complete-splits. The full-speed transaction (for these cases) can deliver data on the full-speed bus segment during micro-frame 7 of $H\text{-}Frame_{Y+1}$, or micro-frame 0 of $H\text{-}Frame_{Y+2}$. The complete splits are scheduled using $siTD_{X+2}$ (not shown). The complete-splits to extract this data must use the buffer pointer from $siTD_{X+1}$. The only way for the host controller to reach $siTD_{X+1}$ from $H\text{-}Frame_{Y+2}$is to use $siTD_{X+2}$'s back pointer. The host controller rules for when to use the back pointer are described is Section Periodic Isochronous - Do Complete Split .

Software must apply the following rules when calculating the schedule and linking the schedule data structures into the periodic schedule:

- Software must ensure that an isochronous split-transaction is started so that it will complete before the end of the *B-Frame*.
- Software must ensure that for a single full-speed isochronous endpoint, there is never a start-split and complete-split in *H-Frame, micro-frame 1*. This is mandated as a rule so that case 2a and case 2b can be discriminated. According to the core USB specification, the long isochronous transaction illustrated in Case 2b, could be scheduled so that the start-split was in micro-frame 1 of *H-Frame* N and the last complete-split would need to occur in micro-frame 1 of H-Frame N+1. However, it is impossible to discriminate between cases 2a and case 2b, which has significant impact on the complexity of the host controller.

65.4.3.12.3.2 Tracking Split Transaction Progress for Isochronous Transfers

To correctly maintain the data stream, the host controller must be able to detect and report errors where device to host data is lost. Isochronous endpoints do not employ the concept of a halt on error, however the host is required to identify and report per-packet errors observed in the data stream. This includes schedule traversal problems (skipped micro-frames), timeouts and corrupted data received.

In similar kind to interrupt split-transactions, the portions of the split transaction protocol must execute in the micro-frames they are scheduled. The queue head data structure used to manage full- and low-speed interrupt has several mechanisms for tracking when portions of a transaction have occurred. Isochronous transfers use siTDs, for their transfers, and the data structures are only reachable via the schedule in the exact micro-frame in which they are required (so all the mechanism employed for tracking in queue heads is not required for siTDs). Software has the option of reusing siTD several times in the complete periodic schedule. However, it must ensure that the results of split transaction N are consumed and the siTD reinitialized (activated) before the host controller gets back to the siTD (in a future micro-frame).

Split-transaction isochronous OUTs utilize a low-level protocol to indicate which portions of the split transaction data have arrived. Control over the low-level protocol is exposed in an siTD via the fields *Transaction Position* (*TP*) and *Transaction Count* (*T-count*). If the entire data payload for the OUT split transaction is larger than 188 bytes, there will be more than one start-split transaction, each of which require proper annotation. If host hold-offs occur, then the sequence of annotations received from the host will not be complete, which is detected and handled by the transaction translator. See Section Periodic Isochronous - Do Start Split for a description on how these fields are used during a sequence of start-split transactions.

The fields *siTD.T-Count* and *siTD.TP* are used by the host controller to drive and sequence the transaction position annotations. It is the responsibility of system software to properly initialize these fields in each siTD. Once the budget for a split-transaction

isochronous endpoint is established, *S-mask*, *T-Count*, and *TP* initialization values for all the siTD associated with the endpoint are constant. They remain constant until the budget for the endpoint is recalculated by software and the periodic schedule adjusted.

For IN-endpoints, the transaction translator simply annotates the response data packets with enough information to allow the host controller to identify the last data. As with split transaction Interrupt, it is the host controller's responsibility to detect when it has missed an opportunity to execute a complete-split. The following field in the siTD is used to track and detect errors in the execution of a split transaction for an IN isochronous endpoint.

- *C-prog-mask*. This is an eight-bit bit-vector where the host controller keeps track of which complete-splits have been executed. Due to the nature of the Transaction Translator periodic pipeline, the complete-splits need to be executed in-order. The host controller needs to detect when the complete-splits have not been executed in order. This can only occur due to system hold-offs where the host controller cannot get to the memory-based schedule. *C-prog-mask* is a simple bit-vector that the host controller sets a bit for each complete-split executed. The bit position is determined by the micro-frame (USB_n_FRINDEX[2:0]) number in which the complete-split was executed. The host controller always checks *C-prog-mask* before executing a complete-split transaction. If the previous complete-splits have not been executed, then it means one (or more) have been skipped and data has potentially been lost. System software is required to initialize this field to zero before setting an siTD's *Active* bit to a one.

If a transaction translator returns with the final data before all of the complete-splits have been executed, the state of the transfer is advanced so that the remaining complete-splits are not executed. Refer to Section Asynchronous - Do Complete Split for a description on how the state of the transfer is advanced. It is important to note that an IN siTD is retired based solely on the responses from the Transaction Translator to the complete-split transactions. This means, for example, that it is possible for a transaction translator to respond to a complete-split with an MDATA PID. The number of bytes in the MDATA's data payload could cause the siTD field *Total Bytes to Transfer* to decrement to zero. This response can occur, before all of the scheduled complete-splits have been executed. In other interface, data structures (for example, high-speed data streams through queue heads), the transition of *Total Bytes to Transfer* to zero signals the end of the transfer and results in setting of the *Active* bit to zero. However, in this case, the result has not been delivered by the Transaction Translator and the host must continue with the next complete-split transaction to extract the residual transaction state. This scenario occurs because of the pipeline rules for a Transaction Translator (see Chapter 11 of the Universal Serial Bus Revision 2.0). In summary the periodic pipeline rules require that on a micro-frame boundary, the Transaction Translator will hold the final two bytes received (if it has not seen an End Of Packet (EOP)) in the full-speed bus pipe stage and give the

remaining bytes to the high-speed pipeline stage. At the micro-frame boundary, the Transaction Translator could have received the entire packet (including both CRC bytes) but not received the packet EOP. In the next micro-frame, the Transaction Translator will respond with an MDATA and send all of the data bytes (with the two CRC bytes being held in the full-speed pipeline stage). This could cause the siTD to decrement its *Total Bytes to Transfer* field to zero, indicating it has received all expected data. The host must still execute one more (scheduled) complete-split transaction in order to extract the results of the full-speed transaction from the Transaction Translator (for example, the Transaction Translator may have detected a CRC failure, and this result must be forwarded to the host).

If the host experiences hold-offs that cause the host controller to skip one or more (but not all) scheduled split transactions for an isochronous OUT, then the protocol to the transaction translator will not be consistent and the transaction translator will detect and react to the problem. Likewise, for host hold-offs that cause the host controller to skip one or more (but not all) scheduled split transactions for an isochronous IN, the *C-prog-mask* is used by the host controller to detect errors. However, if the host experiences a hold-off that causes it to skip all of an siTD, or an siTD expires during a host hold off (for example, a hold-off occurs and the siTD is no longer reachable by the host controller in order for it to report the hold-off event), then system software must detect that the siTDs have not been processed by the host controller (that is, state not advanced) and report the appropriate error to the client driver.

65.4.3.12.3.3 Split Transaction Execution State Machine for Isochronous

In the following presentation, all references to micro-frame are in the context of a micro-frame within an *H-Frame.*

If the *Active* bit in the *Status* byte is a zero, the host controller will ignore the siTD and continue traversing the periodic schedule. Otherwise the host controller will process the siTD as specified below. A split transaction state machine is used to manage the split-transaction protocol sequence. The host controller uses the fields defined in Section Tracking Split Transaction Progress for Interrupt Transfers , plus the variable *cMicroFrameBit* defined in Section Split Transaction Execution State Machine for Interrupt to track the progress of an isochronous split transaction. The figure below illustrates the state machine for managing an siTD through an isochronous split transaction. Bold, dotted circles denote the state of the *Active* bit in the *Status* field of a siTD. The Bold, dotted arcs denote the transitions between these states. Solid circles denote the states of the split transaction state machine and the solid arcs denote the transitions between these states. Dotted arcs and boxes reference actions that take place either as a result of a transition or from being in a state.

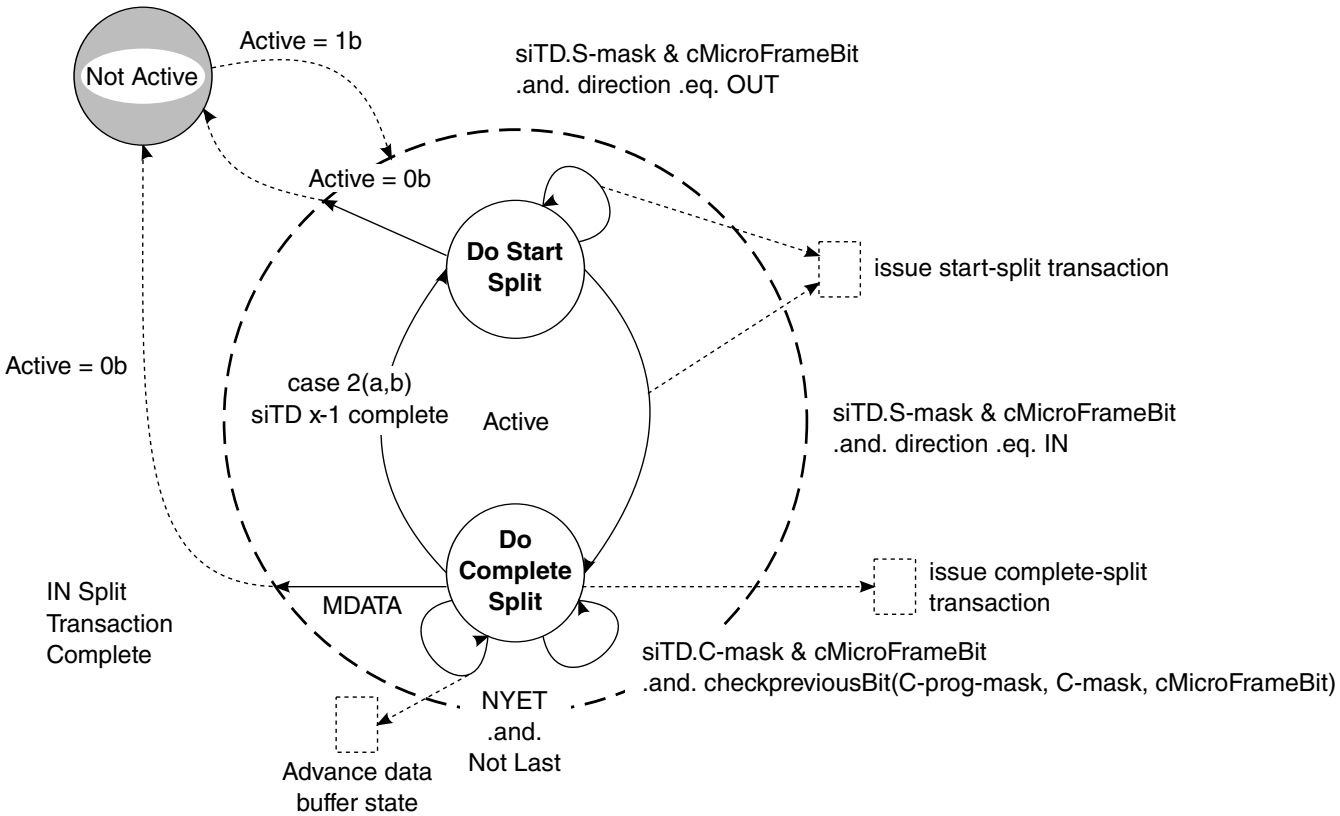


Figure 65-27. Split Transaction State Machine for Isochronous

65.4.3.12.3.4 Periodic Isochronous - Do Start Split

Isochronous split transaction OUTs use only this state.

An siTD for a split-transaction isochronous IN is either initialized to this state, or the siTD transitions to this state from Do Complete Split when a case 2a (IN) or 2b scheduling boundary isochronous split-transaction completes.

Each time the host controller reaches an active siTD in this state, it checks the *siTD.S-mask* against *cMicroFrameBit*. If there is a one in the appropriate position, the siTD will execute a start-split transaction. By definition, the host controller cannot *reach* an siTD at the wrong time. If the *I/O* field indicates an IN, then the start-split transaction includes only the extended token plus the full-speed token. Software must initialize the *siTD.Total Bytes To Transfer* field to the number of bytes expected. This is usually the maximum packet size for the full-speed endpoint. The host controller exits this state when the start-split transaction is complete.

The remainder of this section is specific to an isochronous OUT endpoint (that is, the *I/O* field indicates an OUT). When the host controller executes a start-split transaction for an isochronous OUT it includes a data payload in the start-split transaction. The memory

buffer address for the data payload is constructed by concatenating *siTD.Current Offset* with the page pointer indicated by the page selector field (*siTD.P*). A zero in this field selects Page 0 and a 1 selects Page 1. During the start-split for an OUT, if the data transfer crosses a page boundary during the transaction, the host controller must detect the page cross, update the *siTD.P*-bit from a zero to a one, and begin using the *siTD.Page 1* with *siTD.Current Offset* as the memory address pointer. The field *siTD.TP* is used to annotate each start-split transaction with the indication of which part of the split-transaction data the current payload represents (ALL, BEGIN, MID, END). In all cases the host controller simply uses the value in *siTD.TP* to mark the start-split with the correct transaction position code.

T-Count is always initialized to the number of start-splits for the current frame. *TP* is always initialized to the first required transaction position identifier. The scheduling boundary case (see Figure 65-26) is used to determine the initial value of *TP*. The initial cases are summarized in the following table.

Table 65-48. Initial Conditions for OUT siTD's TP and T-count Fields

Case	T-count	TP	Description
1, 2a	=1	ALL	When the OUT data payload is less than (or equal to) 188 bytes, only one start-split is required to move the data. The one start-split must be marked with an ALL.
1, 2a	!=1	BEGIN	When the OUT data payload is greater than 188 bytes more than one start-split must be used to move the data. The initial start-split must be marked with a BEGIN.

After each start-split transaction is complete, the host controller updates *T-Count* and *TP* appropriately so that the next start-split is correctly annotated.

The table below illustrates all of the *TP* and *T-count* transitions, which must be accomplished by the host controller.

.

Table 65-49. Transaction Position (TP)/Transaction Count (T-Count) Transition Table

TP	T-count next	TP next	Description
ALL	0	N/A	Transition from ALL, to done.
BEGIN	1	END	Transition from BEGIN to END. Occurs when *T-count* starts at 2.
BEGIN	!=1	MID	Transition from BEGIN to MID. Occurs when *T-count* starts at greater than 2.
MID	!=1	MID	*TP* stays at MID while *T-count* is not equal to 1 (that is, greater than 1). This case can occur for any of the scheduling boundary cases where the *T-count* starts greater than 3.
MID	1	END	Transition from MID to END. This case can occur for any of the scheduling boundary cases where the *T-count* starts greater than 2.

The start-split transactions do not receive a handshake from the transaction translator, so the host controller always advances the transfer state in the siTD after the bus transaction is complete. To advance the transfer state the following operations take place:

- The *siTD.Total Bytes To Transfer* and the *siTD.Current Offset* fields are adjusted to reflect the number of bytes transferred.
- The *siTD.P* (page selector) bit is updated appropriately.
- The *siTD.TP* and *siTD.T-count* fields are updated appropriately as defined in Table 65-49.

These fields are then written back to the memory based siTD. The *S-mask* is fixed for the life of the current budget. As mentioned above, *TP* and *T-count* are set specifically in each siTD to reflect the data to be sent from this siTD. Therefore, regardless of the value of *S-mask*, the actual number of start-split transactions depends on *T-count* (or equivalently, *Total Bytes to Transfer*). The host controller must set the *Active* bit to a zero when it detects that all of the schedule data has been sent to the bus. The preferred method is to detect when *T-Count* decrements to zero as a result of a start-split bus transaction. Equivalently, the host controller can detect when *Total Bytes to Transfer* decrements to zero. Either implementation must ensure that if the initial condition is *Total Bytes to Transfer* equal to zero and *T-count* is equal to a one, then the host controller will issue a single start-split, with a zero-length data payload. Software must ensure that *TP*, *T-count* and *Total Bytes to Transfer* are set to deliver the appropriate number of bus transactions from each siTD. An inconsistent combination will yield undefined behavior.

If the host experiences hold-offs that cause the host controller to skip start-split transactions for an OUT transfer, the state of the transfer will not progress appropriately. The transaction translator will observe protocol violations in the arrival of the start-splits for the OUT endpoint (that is, the transaction position annotation will be incorrect as received by the transaction translator).

Example scenarios are described in Section Split Transaction for Isochronous - Processing Examples .

A host controller implementation can optionally track the progress of an OUT split transaction by setting appropriate bits in the *siTD.C-prog-mask* as it executes each scheduled start-split. The *checkPreviousBit*() algorithm defined in Periodic Isochronous - Do Complete Split can be used prior to executing each start-split to determine whether start-splits were skipped. The host controller can use this mechanism to detect missed micro-frames. It can then set the siTD's *Active* bit to zero and stop execution of this siTD. This saves on both memory and high-speed bus bandwidth.

65.4.3.12.3.5 Periodic Isochronous - Do Complete Split

This state is only used by a split-transaction isochronous IN endpoint.

This state is entered unconditionally from the Do Start State after a start-split transaction is executed for an IN endpoint. Each time the host controller visits an siTD in this state, it conducts a number of tests to determine whether it should execute a complete-split transaction. The individual tests are listed below. The sequence they are applied depends on which micro-frame the host controller is currently executing which means that the tests might not be applied until after the siTD referenced from the back pointer has been fetched.

- Test A. *cMicroFrameBit* is bit-wise anded with *siTD.C-mask* field. A non-zero result indicates that software scheduled a complete-split for this endpoint, during this micro-frame. This test is always applied to a newly fetched siTD that is in this state.
- Test B. The *siTD.C-prog-mask* bit vector is checked to determine whether the previous complete splits have been executed. An example algorithm is below (this is slightly different than the algorithm used in Section Periodic Isochronous - Do Complete Split). The sequence in which this test is applied depends on the current value of USB_n_FRINDEX[2:0]. If USB_n_FRINDEX[2:0] is 0 or 1, it is not applied until the back pointer has been used. Otherwise it is applied immediately.

```
Algorithm Boolean CheckPreviousBit(siTD.C-prog-mask, siTD.C-mask, cMicroFrameBit)
Begin
          Boolean rvalue = TRUE;
          previousBit = cMicroFrameBit rotate-right(1)
          -- Bit-wise anding previousBit with C-mask indicates whether there was an intent
          -- to send a complete split in the previous micro-frame. So, if the
          -- 'previous bit' is set in C-mask, check C-prog-mask to make sure it
          -- happened.
          if previousBit bitAND siTD.C-mask then
                    if not (previousBit bitAND siTD.C-prog-mask) then
                              rvalue = FALSE
                    End if
          End if
          Return rvalue
End Algorithm
```

If Test A is true and USB_n_FRINDEX[2:0] is zero or one, then this is a case 2a or 2b scheduling boundary (see Figure 65-25). See Section Periodic Isochronous - Do Complete Split for details in handling this condition.

If Test A and Test B evaluate to true, then the host controller will execute a complete-split transaction using the transfer state of the current siTD. When the host controller commits to executing the complete-split transaction, it updates *QH.C-prog-mask* by bit-ORing with *cMicroFrameBit*. The transfer state is advanced based on the completion status of the complete-split transaction. To advance the transfer state of an IN siTD, the host controller must:

- Decrement the number of bytes received from *siTD.Total Bytes To Transfer,*
- Adjust *siTD.Current Offset* by the number of bytes received,

- Adjust *siTD.P* (page selector) field if the transfer caused the host controller to use the next page pointer, and
- Set any appropriate bits in the *siTD.Status* field, depending on the results of the transaction.

Note that if the host controller encounters a condition where *siTD.Total Bytes To Transfer* is zero, and it receives more data, the host controller must not write the additional data to memory. The *siTD.Status.Active* bit must be set to zero and the *siTD.Status.Babble Detected* bit must be set to a one. The fields *siTD.Total Bytes To Transfer*, *siTD.Current Offset*, and *siTD.P* (page selector) are not required to be updated as a result of this transaction attempt.

The host controller must accept (assuming good data packet CRC and sufficient room in the buffer as indicated by the value of *siTD.Total Bytes To Transfer*) MDATA and DATA0/1 data payloads up to and including 192 bytes. A host controller implementation may optionally set *siTD.Status Active* to a zero and *siTD.Status.Babble Detected* to a one when it receives and MDATA or DATA0/1 with a data payload of more than 192 bytes. The following responses have the noted effects:

- ERR. The full-speed transaction completed with a time-out or bad CRC and this is a reflection of that error to the host. The host controller sets the *ERR* bit in the *siTD.Status* field and sets the *Active* bit to a zero.
- Transaction Error (XactErr). The complete-split transaction encounters a Timeout, CRC16 failure, etc. The *siTD.Status* field *XactErr* field is set to a one and the complete-split transaction must be retried immediately. The host controller must use an internal error counter to count the number of retries as a counter field is not provided in the siTD data structure. The host controller will not retry more than two times. If the host controller exhausts the retries or the end of the micro-frame occurs, the *Active* bit is set to zero.
- DATAx (0 or 1). This response signals that the final data for the split transaction has arrived. The transfer state of the siTD is advanced and the *Active* bit is set to a zero. If the *Bytes To Transfer* field has not decremented to zero (including the reception of the data payload in the DATAx response), then less data than was expected, or allowed for was actually received. This *short packet* event does not set the USBINT status bit in the USBSTS register to a one. The host controller will not detect this condition.
- NYET (and Last). On each NYET response, the host controller also checks to determine whether this is the last complete-split for this split transaction. Last was defined in Section Periodic Isochronous - Do Complete Split . If it is the last complete-split (with a NYET response), then the transfer state of the siTD is not advanced (never received any data) and the *Active* bit is set to a zero. No bits are set in the *Status* field because this is essentially a skipped transaction. The transaction translator must have responded to all the scheduled clompete-splits with NYETs,

meaning that the start-split issued by the host controller was not received. This result should be interpreted by system software as if the transaction was completely skipped. The test for whether this is the last complete split can be performed by XORing C-mask with C-prog-mask. A zero result indicates that all complete-splits have been executed.

- MDATA (and Last). See above description for testing for Last. This can only occur when there is an error condition. Either there has been a babble condition on the full-speed link, which delayed the completion of the full-speed transaction, or software set up the *S-mask* and/or *C-mask*s incorrectly. The host controller must set *XactErr* bit to a one and the *Active* bit is set to a zero.
- NYET (and not Last). See above description for testing for Last. The complete-split transaction received a NYET response from the transaction translator. Do not update any transfer state (except for *C-prog-mask*) and stay in this state.
- MDATA (and not Last). The transaction translator responds with an MDATA when it has partial data for the split transaction. For example, the full-speed transaction data payload spans from micro-frame X to X+1 and during micro-frame X, the transaction translator will respond with an MDATA and the data accumulated up to the end of micro-frame X. The host controller advances the transfer state to reflect the number of bytes received.

If Test A succeeds, but Test B fails, it means that one or more of the complete-splits have been skipped. The host controller sets the *Missed Micro-Frame* status bit and sets the *Active* bit to a zero.

65.4.3.12.3.6 Complete-Split for Scheduling Boundary Cases 2a, 2b

Boundary cases 2a and 2b (INs only) (see Figure 65-25) require that the host controller use the transaction state context of the previous siTD to finish the split transaction. The table below enumerates the transaction state fields.

Table 65-50. Summary siTD Split Transaction State

Buffer State	Status	Execution Progress
Total Bytes To Transfer P (page select) Current Offset TP (transaction position) T-count (transaction count)	All bits in the status field	C-prog-mask

NOTE

TP and *T-count* are used only for Host to Device (OUT) endpoints.

If software has budgeted the schedule of this data stream with a frame wrap case, then it must initialize the *siTD.Back Pointer* field to reference a valid siTD and will have the *siTD.Back Pointer.T-bit* in the *siTD.Back Pointer*

field set to a zero. Otherwise, software must set the *siTD.Back Pointer.T-bit* in the *siTD.Back Pointer* field to a one. The host controller's rules for interpreting when to use the *siTD.Back Pointer* field are listed below. These rules apply only when the siTD's *Active* bit is a one and the *SplitXState* is Do Complete Split.

- When cMicroFrameBit is a 1h and the siTDX.Back Pointer.T-bit is a zero, or
- If cMicroFrameBit is a 2h and siTDX.S-mask[0] is a zero

When either of these conditions apply, then the host controller must use the transaction state from $siTD_{X-1}$.

In order to access $siTD_{X-1}$, the host controller reads on-chip the siTD referenced from $siTD_X$.*Back Pointer*.

The host controller must save the entire state from $siTD_X$ while processing $siTD_{X-1}$. This is to accommodate for case 2b processing. The host controller must not recursively walk the list of *siTD.Back Pointers*.

If $siTD_{X-1}$ is active (*Active* bit is a one and *SplitXStat* is Do Complete Split), then both Test A and Test B are applied as described above. If these criteria to execute a complete-split are met, the host controller executes the complete split and evaluates the results as described above. The transaction state (see Table 65-50) of $siTD_{X-1}$ is appropriately advanced based on the results and written back to memory. If the resultant state of $siTD_{X-1}$'s *Active* bit is a one, then the host controller returns to the context of $siTD_X$, and follows its next pointer to the next schedule item. No updates to $siTD_X$ are necessary.

If $siTD_{X-1}$ is active (*Active* bit is a one and *SplitXStat* is Do Start Split), then the host controller must set *Active* bit to a zero and *Missed Micro-Frame* status bit to a one and the resultant status written back to memory.

If $siTD_{X-1}$*'sActive* bit is a zero, (because it was zero when the host controller first visited $siTD_{X-1}$ via $siTD_X$*'s* back pointer, it transitioned to zero as a result of a detected error, or the results of $siTD_{X-1}$*'s* complete-split transaction transitioned it to zero), then the host controller returns to the context of $siTD_X$ and transitions its *SplitXState* to Do Start Split. The host controller then determines whether the case 2b start split boundary condition exists (that is, if *cMicroframeBit* is a 1b and $siTD_X$.*S-mask*[0] is a 1b). If this criterion is met the host controller immediately executes a start-split transaction and appropriately advances the transaction state of $siTD_X$, then follows $siTD_X$.*Next Pointer* to the next schedule item. If the criterion is not met, the host controller simply follows $siTD_X$.*Next Pointer* to the next schedule item. Note that in the case of a 2b boundary case, the split-transaction of $siTD_{X-1}$ will have its *Active* bit set to zero when the host controller returns

to the context of $siTD_X$. Also, note that software should not initialize an siTD with *C-mask* bits 0 and 1 set to a one and an *S-mask* with bit zero set to a one. This scheduling combination is not supported and the behavior of the host controller is undefined.

65.4.3.12.3.7 Split Transaction for Isochronous - Processing Examples

There is an important difference between how the hardware/software manages the isochronous split transaction state machine and how it manages the asynchronous and interrupt split transaction state machines.

The asynchronous and interrupt split transaction state machines are encapsulated within a single queue head. The progress of the data stream depends on the progress of each split transaction. In some respects, the split-transaction state machine is sequenced via the Execute Transaction queue head traversal state machine (see Figure 65-18).

Isochronous is a pure time-oriented transaction/data stream. The interface data structures are optimized to efficiently describe transactions that need to occur at specific times. The isochronous split-transaction state machine must be managed across these time-oriented data structures. This means that system software must correctly describe the scheduling of split-transactions across more than one data structure.

Then the host controller must make the appropriate state transitions at the appropriate times, in the correct data structures.

For example, the table below illustrates a couple of frames worth of scheduling required to schedule a case 2a full-speed isochronous data stream.

Table 65-51. Example Case 2a - Software Scheduling siTDs for an IN Endpoint

siTDX		Micro-Frames								Initial
#	Masks	0	1	2	3	4	5	6	7	SplitXState
X	S-Mask	-	-	-	-	1	-	-	-	Do Start Split
	C-Mask	1	1	-	-	-	-	1	1	
X+1	S-Mask	-	-	-	-	1	-	-	-	Do Complete Split
	C-Mask	1	1					1	1	
X+2	S-Mask	-	-	-	-	1	-	-	-	Do Complete Split
	C-Mask	1	1					1	1	
X+3	S-Mask	Repeats previous pattern								Do Complete Split
	C-Mask									

This example shows the first three siTDs for the transaction stream. Because this is the case-2a frame-wrap case, *S-masks* of all siTDs for this endpoint have a value of 10h (a one bit in micro-frame 4) and *C-mask* value of C3h (one-bits in micro-frames 0,1, 6 and 7). Additionally, sofware ensures that the *Back Pointer* field of each siTD references the appropriate siTD data structure (and the *Back PointerT-bits* are set to zero).

The initial *SplitXState* of the first siTD is Do Start Split. The host controller will visit the first siTD eight times during frame X. The C-mask bits in micro-frames 0 and 1 are ignored because the state is Do Start Split. During micro-frame 4, the host controller determines that it can run a start-split (and does) and changes *SplitXState* to Do Complete Split. During micro-frames 6 and 7, the host controller executes complete-splits. Notice the siTD for frame X+1 has its *SplitXState* initialized to Do Complete Split. As the host controller continues to traverse the schedule during *H-Frame* X+1, it will visit the second siTD eight times. During micro-frames 0 and 1 it will detect that it must execute complete-splits.

During *H-Frame* X+1, micro-frame 0, the host controller detects that $siTD_{X+1}$'s *Back Pointer.T-bit* is a zero, saves the state of $siTD_{X+1}$ and fetches $siTD_X$. It executes the complete split transaction using the transaction state of $siTD_X$. If the $siTD_X$ split transaction is complete, siTD's *Active* bit is set to zero and results written back to $siTD_X$. The host controller retains the fact that $siTD_X$ is retired and transitions the *SplitXState* in the $siTD_{X+1}$ to Do Start Split. At this point, the host controller is prepared to execute the start-split for $siTD_{X+1}$ when it reaches micro-frame 4. If the split-transaction completes early (transaction-complete is defined in Section Periodic Isochronous - Do Complete Split), that is, before all the scheduled complete-splits have been executed, the host controller will transition $siTD_X$.*SplitXState* to Do Start Split early and naturally skip the remaining scheduled complete-split transactions. For this example, $siTD_{X+1}$ does not receive a DATA0 response until *H-Frame* X+2, micro-frame 1.

During *H-Frame X+2,* micro-frame 0, the host controller detects that $siTD_{X+2}$'s *Back Pointer.T-bit* is a zero, saves the state of $siTD_{X+2}$ and fetches $siTD_{X+1}$. As described above, it executes another split transaction, receives an MDATA response, updates the transfer state, but does not modify the *Active* bit. The host controller returns to the context of $siTD_{X+2}$, and traverses its next pointer without any state change updates to $siTD_{X+2}$. S

During *H-Frame X+2*, micro-frame 1, the host controller detects $siTD_{X+2}$'s *S-mask[0]* is a zero, saves the state of $siTD_{X+2}$ and fetches $siTD_{X+1}$. It executes another complete-split transaction, receives a DATA0 response, updates the transfer state and sets the *Active* bit to a zero. It returns to the state of $siTD_{X+2}$ and changes its *SplitXState* to Do Start Split. At this point, the host controller is prepared to execute start-splits for $siTD_{X+2}$ when it

reaches micro-frame 4. <TBD… describe how software detects that there was missing micro-frames (don't think we care about missing out micro-frames. There is enough residual state to identify than not all transactions were executed.).

65.4.3.13 Host Controller Pause

When the host controller's *HCHalted* bit in the USBSTS register is a zero, the host controller is sending SOF (Start OF Frame) packets down all enabled ports.

When the schedules are enabled, the EHCI host controller will access the schedules in main memory each micro-frame. This constant pinging of main memory is known to create ARM platform power management problems for mobile systems. Specifically, mobile systems aggressively manage the state of the ARM platform, based on recent history usage. In the more aggressive power saving modes, the ARM platform can disable its caches. Current PC architectures assume that bus-master accesses to main memory must be cache-coherent. So, when bus masters are busy touching memory, the ARM platform power management software can detect this activity over time and inhibit the transition of the ARM platform into its lowest power savings mode. USB controllers are bus-masters and the frequency at which they access their memory-based schedules keeps the ARM platform power management software from placing the ARM platform into its lowest power savings state.

USB Host controllers don't access main memory when they are suspended. However, there are a variety of reasons why placing the USB controllers into suspend won't work, but they are beyond the scope of this document. The base requirement is that the USB controller needs to be kept out of main memory, while at the same time, the USB bus is kept from going into suspend.

EHCI controllers provide a large-grained mechanism that can be manipulated by system software to change the memory access pattern of the host controller. System software can manipulate the schedule enable bits in the USBCMD register to turn on/off the scheduling traversal. A software heuristic can be applied to implement an on/off duty cycle that allows the USB to make reasonable progress and allow the ARM platform power management to get the ARM platform into its lowest power state. This method is not intended to be applied at all times to throttle USB, but should only be applied in very specific configurations and usage loads. For example, when only a keyboard or mouse is attached to the USB, the heuristic could detect times when the USB is attempting to move data only very infrequently and can adjust the duty cycle to allow the ARM platform to reach its low power state for longer periods of time. Similarly, it could detect increases in the USB load and adjust the duty cycle appropriately, even to the point where the schedules are never disabled. The assumption here is that the USB is moving data and the ARM platform will be required to process the data streams.

It is suggested that in order to provide a complete solution for the system, the companion host controllers should also provide a similar method to allow system software to inhibit the companion host controller from accessing its shared memory based data structures (schedule lists or otherwise).

65.4.3.14 Port Test Modes -Host Operational Model

EHCI host controllers must implement the port test modes Test J_State, Test K_State, Test_Packet, Test Force_Enable, and Test SE0_NAK as described in the USB Specification Revision 2.0.

The system is only allowed to test ports that are owned by the EHCI controller (for example, *CF-bit* is a one and *PortOwner* bit is a zero). System software is allowed to have at most one port in test mode at a time. Placing more than one port in test mode will yield undefined results. The required, per port test sequence is (assuming the *CF-bit* in the USB_n_CONFIGFLAG register is a one):

- Disable the periodic and asynchronous schedules by setting the *Asynchronous Schedule Enable* and *Periodic Schedule Enable* bits in the USBCMD register to a zero.
- Place all enabled root ports into the suspended state by setting the *Suspend* bit in each appropriate USB_n_PORTSC register to a one.
- Set the *Run/Stop* bit in the USBCMD register to a zero and wait for the *HCHalted* bit in the USBSTS register, to transition to a one. Note that an EHCI host controller implementation may optionally allow port testing with the *Run/Stop* bit set to a one. However, all host controllers must support port testing with *Run/Stop* set to a zero and *HCHalted* set to a one.
- Set the *Port Test Control* field in the port under test PORTSC register to the value corresponding to the desired test mode. If the selected test is Test_Force_Enable, then the *Run/Stop* bit in the USBCMD register must then be transitioned back to one, in order to enable transmission of SOFs out of the port under test.
- When the test is complete, system software must ensure the host controller is halted (*HCHalted* bit is a one) then it terminates and exits test mode by setting *HCReset* to a one.

65.4.3.15 Interrupts-Host Operational Model

The EHCI Host Controller hardware provides interrupt capability based on a number of sources.

There are several general groups of interrupt sources:

- Interrupts as a result of executing transactions from the schedule (success and error conditions),
- Host controller events (Port change events, etc.), and
- Host Controller error events

All transaction-based sources are maskable through the Host Controller's Interrupt Enable register (USBINTR, see Section Interrupt Enable Register (USBC_*n*_USBINTR)). Additionally, individual transfer descriptors can be marked to generate an interrupt on completion. This section describes each interrupt source and the processing that occurs in response to the interrupt.

During normal operation, interrupts may be immediate or deferred until the next interrupt threshold occurs. The interrupt threshold is a tunable parameter via the *Interrupt Threshold Control* field in the USBCMD register. The value of this register controls when the host controller will generate an interrupt on behalf of normal transaction execution. When a transaction completes during an interrupt interval period, the interrupt signaling the completion of the transfer will not occur until the interrupt threshold occurs. For example, the default value is eight micro-frames. This means that the host controller will not generate interrupts any more frequently than once every eight micro-frames.

Section Host System Error details effects of a host system error.

If an interrupt has been scheduled to be generated for the current interrupt threshold interval, the interrupt is not signaled until after the status for the last complete transaction in the interval has been written back to host memory. This may sometimes result in the interrupt not being signaled until the next interrupt threshold.

Initial interrupt processing is the same, regardless of the reason for the interrupt. When an interrupt is signaled by the hardware, ARM platform control is transferred to host controller's USB interrupt handler. The precise mechanism to accomplish the transfer is OS specific. For this discussion it is just assumed that control is received. When the interrupt handler receives control, its first action is to reads the USBSTS (USB Status Register). It then acknowledges the interrupt by clearing all of the interrupt status bits by writing ones to these bit positions. The handler then determines whether the interrupt is due to schedule processing or some other event. After acknowledging the interrupt, the handler (via an OS-specific mechanism), schedules a deferred procedure call (DPC) which will execute later. The DPC routine processes the results of the schedule execution. The precise mechanisms used are beyond the scope of this document.

Note: the host controller is not required to de-assert a currently active interrupt condition when software sets the interrupt enables (in the USBINR register, see Section Interrupt Enable Register (USBC_*n*_USBINTR)) to a zero. The only reliable method software

should use for acknowledging an interrupt is by transitioning the appropriate status bits in the USBSTS register (Section USB Status Register (USBC_*n*_USBSTS)) from a one to a zero.

65.4.3.15.1 Transfer/Transaction Based Interrupts

These interrupt sources are associated with transfer and transaction progress. They are all dependent on the next interrupt threshold.

65.4.3.15.1.1 Transaction Error

A transaction error is any error that caused the host controller to think that the transfer did not complete successfully.

The table below lists the events/responses that the host can observe as a result of a transaction. The effects of the error counter and interrupt status are summarized in the following paragraphs. Most of these errors set the *XactErr* status bit in the appropriate interface data structure.

There is a small set of protocol errors that relate only when executing a queue head and fit under the umbrella of a WRONG PID error that are significant to explicitly identify. When these errors occur, the *XactErr* status bit in the queue head is set and the *CErr* field is decremented. When the *PIDCode* indicates a SETUP, the following responses are protocol errors and result in *XactErr* bit being set to a one and the *CErr* field being decremented.

- *EPS* field indicates a high-speed device and it returns a Nak handshake to a SETUP.
- *EPS* field indicates a high-speed device and it returns a Nyet handshake to a SETUP.
- *EPS* field indicates a low- or full-speed device and the complete-split receives a Nak handshake.

Table 65-52. Summary of Transaction Errors

Event / Result	Queue Head/qTD/iTD/siTD Side-effects		USB Status Register (USBSTS)
	Cerr	Status Field	USBERRINT
CRC	-1	XactErr set to a one.	1[1]
Timeout	-1	XactErr set to a one.	1[1]
Bad PID[2]	-1	XactErr set to a one.	1[1]
Babble	N/A	Section Serial Bus Babble	1
Buffer Error	N/A	Section Data Buffer Error	

1. If occurs in a queue head, then *USBERRINT* is asserted only when *CErr* counts down from a one to a zero. In addition the queue is halted, see Halting a Queue Head.
2. The host controller received a response from the device, but it could not recognize the PID as a valid PID.

65.4.3.15.1.2 Serial Bus Babble

When a device transmits more data on the USB than the host controller is expecting for this transaction, it is defined to be babbling. In general, this is called a *Packet Babble*.

When a device sends more data than the *Maximum Length* number of bytes, the host controller sets the *Babble Detected* bit to a one and halts the endpoint if it is using a queue head (see Halting a Queue Head). *Maximum Length* is defined as the minimum of *Total Bytes to Transfer* and *Maximum Packet Size*. The *CErr* field is not decremented for a packet babble condition (only applies to queue heads). A babble condition also exists if IN transaction is in progress at High-speed EOF2 point. This is called a frame babble. A frame babble condition is recorded into the appropriate schedule data structure. In addition, the host controller must disable the port to which the frame babble is detected.

The *USBERRINT* bit in the USB_n_USBSTS register is set to a one and if the *USB Error Interrupt Enable* bit in the USB_n_USBINTR register is a one, then a hardware interrupt is signaled to the system at the next interrupt threshold. The host controller must never start an OUT transaction that will babble across a micro-frame EOF.

NOTE

When a host controller detects a data PID mismatch, it must either: disable the packet babble checking for the duration of the bus transaction or do packet babble checking based solely on *Maximum Packet Size*. The USB core specification defines the requirements on a data receiver when it receives a data PID mismatch (for example, expects a DATA0 and gets a DATA1 or visa-versa). In summary, it must ignore the received data and respond with an ACK handshake, in order to advance the transmitter's data sequence.

The EHCI interface allows System software to provide buffers for a Control, Bulk or Interrupt IN endpoint that are not an even multiple of the maximum packet size specified by the device. Whenever a device misses an ACK for an IN endpoint, the host and device are out of synchronization with respect to the progress of the data transfer. The host controller may have advanced the transfer to a buffer that is less than maximum packet size. The device will re-send its maximum packet size data packet, with the original data PID, in response to the next IN token. In order to properly manage the bus protocol, the host controller must disable the packet babble check when it observes the data PID mismatch.

65.4.3.15.1.3 Data Buffer Error

This event indicates that an overrun of incoming data or a underrun of outgoing data has occurred for this transaction.

This would generally be caused by the host controller not being able to access required data buffers in memory within necessary latency requirements. These conditions are not considered transaction errors, and do not effect the error count in the queue head. When these errors do occur, the host controller records the fact the error occurred by setting the *Data Buffer Error* bit in the queue head, iTD or siTD.

If the data buffer error occurs on a non-isochronous IN, the host controller will not issue a handshake to the endpoint. This will force the endpoint to resend the same data (and data toggle) in response to the next IN to the endpoint.

If the data buffer error occurs on an OUT, the host controller must corrupt the end of the packet so that it cannot be interpreted by the device as a good data packet. Simply truncating the packet is not considered acceptable. An acceptable implementation option is to 1's complement the CRC bytes and send them. There are other options suggested in the Transaction Translator section of the USB Specification Revision 2.0.

65.4.3.15.1.4 USB Interrupt (Interrupt on Completion (IOC))

Transfer Descriptors (iTDs, siTDs, and queue heads (qTDs)) contain a bit that can be set to cause an interrupt on their completion. The completion of the transfer associated with that schedule item causes the USB Interrupt (USBINT) bit in the USB_n_USBSTS register to be set to a one.

In addition, if a short packet is encountered on an IN transaction associated with a queue head, then this event also causes USBINT to be set to a one. If the USB Interrupt Enable bit in the USB_n_USBINTR register is set to a one, a hardware interrupt is signaled to the system at the next interrupt threshold. If the completion is because of errors, the *USBERRINT* bit in the USB_n_USBSTS register is also set to a one.

65.4.3.15.1.5 Short Packet

Reception of a data packet that is less than the endpoint's Max Packet size during Control, Bulk or Interrupt transfers signals the completion of the transfer. Whenever a short packet completion occurs during a queue head execution, the *USBINT* bit in the USB_n_USBSTS register is set to a one.

If the *USB Interrupt Enable* bit is set in the USB_n_USBINTR register, a hardware interrupt is signaled to the system at the next interrupt threshold.

65.4.3.15.2 Host Controller Event Interrupts

These interrupt sources are independent of the interrupt threshold (with the one exception being the Interrupt on Async Advance, see Section Interrupt on Async Advance).

65.4.3.15.2.1 Port Change Events

Port registers contain status and status change bits. When the status change bits are set to a one, the host controller sets the *Port Change Detect* bit in the USBSTS register to a one.

If the *Port Change Interrupt Enable* bit in the USB_n_USBINTR register is a one, then the host controller will issue a hardware interrupt. The port status change bits include:

- Connect Status Change
- Port Enable/Disable Change
- Over-current Change
- Force Port Resume

65.4.3.15.2.2 Frame List Rollover

This event indicates that the host controller has wrapped the frame list. The current programmed size of the frame list effects how often this interrupt occurs.

If the frame list size is 1024, then the interrupt will occur every 1024 milliseconds, if it is 512, then it will occur every 512 milliseconds, etc. When a frame list rollover is detected, the host controller sets the *Frame List Rollover* bit in the USB.USBSTS register to a one. If the *Frame List Rollover Enable* bit in the USB.USBINTR register is set to a one, the host controller issues a hardware interrupt. This interrupt is not delayed to the next interrupt threshold.

65.4.3.15.2.3 Interrupt on Async Advance

This event is used for deterministic removal of queue heads from the asynchronous schedule. Whenever the host controller advances the on-chip context of the asynchronous schedule, it evaluates the value of the *Interrupt on Async Advance Doorbell* bit in the USB.USBCMD register.

If it is a one, it sets the *Interrupt on Async Advance* bit in the USB.USBSTS register to a one. If the *Interrupton Async Advance Enable* bit in the USB.USBINTR register is a one, the host controller issues a hardware interrupt at the next interrupt threshold. A detailed explanation of this feature is described in Section Removing Queue Heads from Asynchronous Schedule .

65.4.3.15.2.4 Host System Error

The host controller is a bus master and any interaction between the host controller and the system may experience errors.

The type of host error may be catastrophic to the host controller (such as a Master Abort) making it impossible for the host controller to continue in a coherent fashion.In the presence of non-catastrophic host errors, such as parity errors, the host controller could potentially continue operation. The recommended behavior for these types of errors is to escalate it to a catastrophic error and halt the host controller. Host-based error must result in the following actions:

- The *Run/Stop* bit in the USB.USBCMD register is set to a zero.
- The following bits in the USB.USBSTS register are set:
  - *Host System Error* bit is to a one.
  - *HCHalted* bit is set to a one.
- If the *Host System Error Enable* bit in the USB.USBINTR register is a one, then the host controller will issue a hardware interrupt. This interrupt is not delayed to the next interrupt threshold. The following table summarizes the required actions taken on the various host errors.

Table 65-53. Summary Behavior of EHCI Host Controller on Host System Errors

Cycle Type	Master Abort	Target Abort	Data Phase Parity
Frame list pointer fetch (read)	Fatal	Fatal	Fatal [o]
siTD fetch (read)	Fatal	Fatal	Fatal [o]
siTD status write-back (write)	Fatal [o]	Fatal [o]	Fatal [o]
iTD fetch (read)	Fatal	Fatal	Fatal [o]
iTD status write-back (write)	Fatal [o]	Fatal [o]	Fatal [o]
qTD fetch (read)	Fatal	Fatal	Fatal [o]
qHD status write-back (write)	Fatal [o]	Fatal [o]	Fatal [o]
Data write	Fatal [o]	Fatal [o]	Fatal [o]
Data read	Fatal	Fatal	Fatal [o]

Potentially, a host controller implementation could continue operation without a halt. However, the recommended behavior is to halt the host controller.

NOTE

After a *Host System Error*, Software must reset the host controller through *HCReset* in the USB.USBCMD register before re-initializing and restarting the host controller.

65.4.4 EHCI Deviation

For the purposes a dual-role Host/Device controller with support for On-The-Go applications, it is necessary to deviate from the EHCI specificationEnhanced Host Controller Interface Specification for Universal Serial Bus, Revision 0.95, November 2000, Intel Corporation. http://www.intel.com. Device operation & On-The-Go operation is not specified in the EHCI and thus the implementation supported in this core is proprietary.

The host mode operation of the core is near EHCI compatible with few minor differences documented in this section.

The particulars of the deviations occur in the areas summarized here:

- Embedded Transaction Translator - Allows direct attachment of FS and LS devices in host mode without the need for a companion controller.
- Device operation - In host mode the device operational registers are generally disabled and thus device mode is mostly transparent when in host mode. However, there are a couple exceptions documented in the following sections.
- Embedded design interface - This core does not have a PCI Interface and therefore the PCI configuration registers described in the EHCI specification are not applicable.
- On-The-Go Operation - This design includes an On-The-Go controller for Port #1.

65.4.4.1 Embedded Transaction Translator Function

The USB-HS OTG High-Speed USB On-The-Go OTG controller supports directly connected full and low speed devices without requiring a companion controller by including the capabilities of a USB 2.0 high speed hub transaction translator.

Although there is no separate Transaction Translator block in the system, the transaction translator function normally associated with a high speed hub has been implemented within the DMA and Protocol engine blocks. The embedded transaction translator function is an extension to EHCI interface, but makes use of the standard data structures and operational models that exist in the EHCI specification to support full and low speed devices.

65.4.4.1.1 Capability Registers

The following additions have been added to the capability registers to support the embedded Transaction Translator Function:

- N_TT added to USB.HCSPARAMS - Host Control Structural Parameters
- N_PTT added to USB.HCSPARAMS - Host Control Structural Parameters

65.4.4.1.2 Operational Registers

The following additions have been added to the operational registers to support the embedded TT:

- Addition of two-bit Port Speed (PSPD) to the Port Status & Control (USBC_*n*_PORTSC1) register.

65.4.4.1.3 Discovery-EHCI Deviation

In a standard EHCI controller design, the EHCI host controller driver detects a Full speed (FS) or Low speed (LS) device by noting if the port enable bit is set after the port reset operation.

The port enable will only be set in a standard EHCI controller implementation after the port reset operation and when the host and device negotiate a High-Speed connection (that is, Chirp completes successfully).

Because this controller has an embedded Transaction Translator, the port enable will always be set after the port reset operation regardless of the result of the host device chirp result and the resulting port speed will be indicated by the PSPD field in USB.PORTSCx.

Therefore, the standard EHCI host controller driver requires an alteration to handle directly connected Full and Low speed devices or hubs.

The change is a fundamental one in that is summarized in the following table.

Table 65-54. Summary of EHCI

Standard EHCI	EHCI with embedded Transaction Translator
After port enable bit is set following a connection and reset sequence, the device/hub is assumed to be HS.	After port enable bit is set following a connection and reset sequence, the device/hub speed is noted from USB.PORTSCx.
FS and LS devices are assumed to be downstream from a HS hub thus, all port-level control is performed through the Hub Class to the nearest Hub.	FS and LS device can be either downstream from a HS hub or directly attached. When the FS/LS device is downstream from a HS hub, then port-level control is done using the Hub Class through the nearest Hub. When a FS/LS device is directly attached, then port-level control is accomplished using USB.PORTSCx.
FS and LS devices are assumed to be downstream from a HS hub with HubAddr=X. [where HubAddr > 0 and HubAddr is the address of the Hub where the bus transitions from HS to FS/LS (ie. Split target hub)]	FS and LS device can be either downstream from a HS hub with HubAddr = X [HubAddr > 0] or directly attached [where HubAddr = 0 and HubAddr is the address of the Root Hub where the bus transitions from HS to FS/LS (ie. Split target hub is the root hub)]

65.4.4.1.4 Data Structures

The same data structures used for FS/LS transactions though a HS hub are also used for transactions through the Root Hub with sm embedded Transaction Translator.

Here it is demonstrated how the Hub Address and Endpoint Speed fields should be set for directly attached FS/LS devices and hubs:

1. QH (for direct attach FS/LS) - Async. (Bulk/Control Endpoints) Periodic (Interrupt)
   - Hub Address = 0
   - Transactions to direct attached device/hub.
     - QH.EPS = Port Speed
   - Transactions to a device downstream from direct attached FS hub.
     - QH.EPS = Downstream Device Speed

NOTE

When QH.EPS = 01 (LS) and PORTSCx.PSPD = 00 (FS), a LS-pre-pid will be sent before the transmitting LS traffic.

Maximum Packet Size must be less than or equal 64 or undefined behaviour may result.

2. siTD (for direct attach FS) - Periodic (ISO Endpoint)
   - All FS ISO transactions:
     - Hub Address = 0
     - siTD.EPS = 00 (full speed)
       - Maximum Packet Size must less than or equal to 1023 or undefined behaviour may result.

65.4.4.1.5 Operational Model

The operational models are well defined for the behavior of the Transaction Translator (see USB 2.0 specificationUniversal Serial Bus Specification, Revision 2.0, April 2000, Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, Philips. http://www.usb.org) and for the EHCI controller moving packets between system memory and a USB-HS hub. Because the embedded Transaction Translator exists within the host controller there is no physical bus between EHCI host controller driver and the USB FS/LS bus. These sections will briefly discuss the operational model for how the EHCI and Transaction Translator operational models are combined without the physical bus between. The following sections assume the reader is familiar with both the EHCI and USB 2.0 Transaction Translator operational models.

65.4.4.1.5.1 Micro- frame Pipeline

The EHCI operational model uses the concept of H-frames and B-frames to describe the pipeline between the Host (H) and the Bus (B). The embedded Transaction Translator shall use the same pipeline algorithms specified in the USB 2.0 specification for a Hub-based Transaction Translator.

All periodic transfers always begin at B-frame 0 (after SOF) and continue until the stored periodic transfers are complete. As an example of the micro-frame pipeline implemented in the embedded Transaction Translator, all periodic transfers that are tagged in EHCI to execute in H-frame 0 will be ready to execute on the bus in B-frame 0.

It is important to note that when programming the S-mask and C-masks in the EHCI data structures to schedule periodic transfers for the embedded Transaction Translator, the EHCI host controller driver must follow the same rules specified in EHCI for programming the S-mask and C-mask for downstream Hub-based Transaction Translators.

Once periodic transfers are exhausted, any stored asynchronous transfer will be moved. Asynchronous transfers are opportunistic in that they shall execute whenever possible and their operation is not tied to H-frame and B-frame boundaries with the exception that an asynchronous transfer can not babble through the SOF (start of B-frame 0.)

65.4.4.1.5.2 Split State Machines

The start and complete split operational model differs from EHCI slightly because there is no bus medium between the EHCI controller and the embedded Transaction Translator.

Where a start or complete-split operation would occur by requesting the split to the HS hub, the start/complete split operation is simple an internal operation to the embedded Transaction Translator. The following table summarizes the conditions where handshakes are emulated from internal state instead of actual handshakes to HS split bus traffic.

Table 65-55. Summary of the Conditons of Handshakes[1]

Condition	Emulate TT Response
Start-Split: All asynchronous buffers full.	NAK
Start-Split: All periodic buffers full.	ERR
Start-Split: Success for start of Async. Transaction.	ACK
Start-Split: Start Periodic Transaction.	No Handshake (Ok)
Complete-Split: Failed to find transaction in queue.	Bus Time Out
Complete-Split: Transaction in Queue is Busy.	NYET
Complete-Split: Transaction in Queue is Complete.	[Actual Handshake from LS/FS device]

1. The un-shaded cells represent Start-Splits and the shaded cells represent Complete-Splits

65.4.4.1.5.3 Asynchronous Transaction Scheduling and Buffer Management

The following USB 2.0 specification items are implemented in the embedded Transaction Translator:

65.4.4.1.5.4 USB 2.0 - 11.17.3

- Sequencing is provided & a packet length estimator ensures no full-speed/low-speed packet babbles into SOF time.

65.4.4.1.5.5 USB 2.0 - 11.17.4

- Transaction tracking for 2 data pipes.

65.4.4.1.5.6 Periodic Transaction Scheduling and Buffer Management

The following USB 2.0 specification items are implemented in the embedded Transaction Translator:

65.4.4.1.5.7 USB 2.0 - 11.18.6.[1-2]

- Abort of pending start-splits
  - EOF (and not started in micro-frames 6)
  - Idle for more than 4 micro-frames
- Abort of pending complete-splits
  - EOF
  - Idle for more than 4 micro-frames

65.4.4.1.5.8 USB 2.0 - 11.18.[7-8]

- Transaction tracking for up to 16 data pipes.
- Complete-split transaction searching.

NOTE

There is no data schedule mechanism for these transactions other than the micro-frame pipeline. The embedded TT assumes the number of packets scheduled in a frame does not exceed the frame duration (1 ms) or else undefined behavior may result.

65.4.4.1.5.9 Multiple Transaction Translators

The maximum number of embedded Transaction Translators that is currently supported is one as indicated by the N_TT field in the Host Controller Structural Parameters (USBC_*n*_HCSPARAMS) register.

65.4.4.2 Device Operation

The co-existence of a device operational controller within the host controller has little effect on EHCI compatibility for host operation except as noted in this section.

65.4.4.3 USB_USBMODE Register

Given that the dual-role controller is initialized in neither host nor device mode, the USB Device Mode (USBC_*n*_USBMODE) register must be programmed for host operation before the EHCI host controller driver can begin EHCI host operations.

65.4.4.3.1 Non-Zero Fields the Register File

Some of the reserved fields and reserved addresses in the capability registers and operational register have use in device mode, the following must be adhered to:

- Write operations to all EHCI reserved fields (some of which are device fields) with the operation registers should always be written to zero. This is an EHCI requirement of the device controller driver that must be adhered to.
- Read operations by the host controller must properly mask EHCI reserved fields (some of which are device fields) because fields that are used exclusive for device are undefined in host mode .

65.4.4.3.2 SOF Interrupt

This SOF Interrupt used for device mode is shared as a free running 125us interrupt for host mode.

EHCI does not specify this interrupt but it has been added for convenience and as a potential software time base. See USB Status Register (USBC_*n*_USBSTS) and Interrupt Enable Register (USBC_*n*_USBINTR) registers.

65.4.4.4 Embedded Design Interface

This is an Embedded USB Host Controller as defined by the EHCI specification and thus does not implement the PCI configuration registers.

65.4.4.4.1 Frame Adjust Register

Given that the optional PCI configuration registers are not included in this implementation, there is no corresponding bit level timing adjustments like is provided by the Frame Adjust register in the PCI configuration registers. Starts of micro-frames are timed precisely to 125 us using the transceiver clock as a reference clock. That is, a 60 Mhz transceiver clock for 8-bit physical interfaces & full-speed serial interfaces or 30 Mhz transceiver clock for 16-bit physical interfaces.

65.4.4.5 Miscellaneous variations from EHCI

65.4.4.5.1 Programmable Physical Interface Behaviour

This design supports multiple Physical interfaces which can operate in differing modes when the core is configured with software programmable Physical Interface Modes.

Software programmability allows the selection of the Physical interface part during the board design phase instead of during the chip design phase.

The control bits for selecting the Physical Interface operating mode have been added to the Port Status & Control (USBC_*n*_PORTSC1) register providing a capability that is not defined by EHCI.

65.4.4.5.2 Discovery

65.4.4.5.2.1 Port Reset

The port connect methods specified by EHCI require setting the port reset bit in the Port Status & Control (USBC_*n*_PORTSC1) register for a duration of 10ms. Due to the complexity required to support the attachment of devices that are not high speed there are counter already present in the design that can count the 10ms reset pulse to alleviate the requirement of the software to measure this duration. Therefore, the basic connection is then summarized as the following:

- [Port Change Interrupt] Port connect change occurs to notify the host controller driver that a device has attached.
- Software shall write a '1' to the reset the device.

- Software shall write a '0' to the reset the device after 10 ms.
  - This step, which is necessary in a standard EHCI design, may be omitted with this implementation. Should the EHCI host controller driver attempt to write a '0' to the reset bit while a reset is in progress, the write will simple be ignored and the reset will continue until completion.
- [Port Change Interrupt] Port enable change occurs to notify the host controller that the device in now operational and at this point the port speed has been determined.

65.4.4.5.2.2 Port Speed Detection

After the port change interrupt indicates that a port is enabled, the EHCI stack should determine the port speed. Unlike the EHCI implementation, which will re-assign the port owner for any device that does not connect at High-Speed, this host controller supports direct attach of non High-Speed devices.

Therefore, the following differences are important regarding port speed detection:

- Port Owner is read-only and always reads 0.
- A 2-bit Port Speed indicator has been added to PORTSC to provide the current operating speed of the port to the host controller driver.
- A 1-bit High Speed indicator has been added to PORTSC to signify that the port is in High-Speed vs. Full/Low Speed - *This information is redundant with the 2-bit Port Speed indicator above.*

65.4.4.5.3 Port Test Mode

Port Test Control mode behaves fully as described in EHCI. An alternate host controller driver procedure is not necessary or supported.

65.4.5 Device Data Structures

This section defines the interface data structures used to communicate control, status, and data between Device Controller Driver (DCD) Software and the Device Controller.

The data structure definitions in this chapter support a 32-bit memory buffer address space. The interface consists of device Queue Heads and Transfer Descriptors.

NOTE

Software must ensure that no interface data structure reachable by the Device Controller spans a 4K-page boundary.

The data structures defined in the chapter are (from the device controller's perspective) a mix of read-only and read/ writable fields. The device controller must preserve the read-only fields on all data structure writes.

The figure below shows the organization of the EndPoint Queue Head.

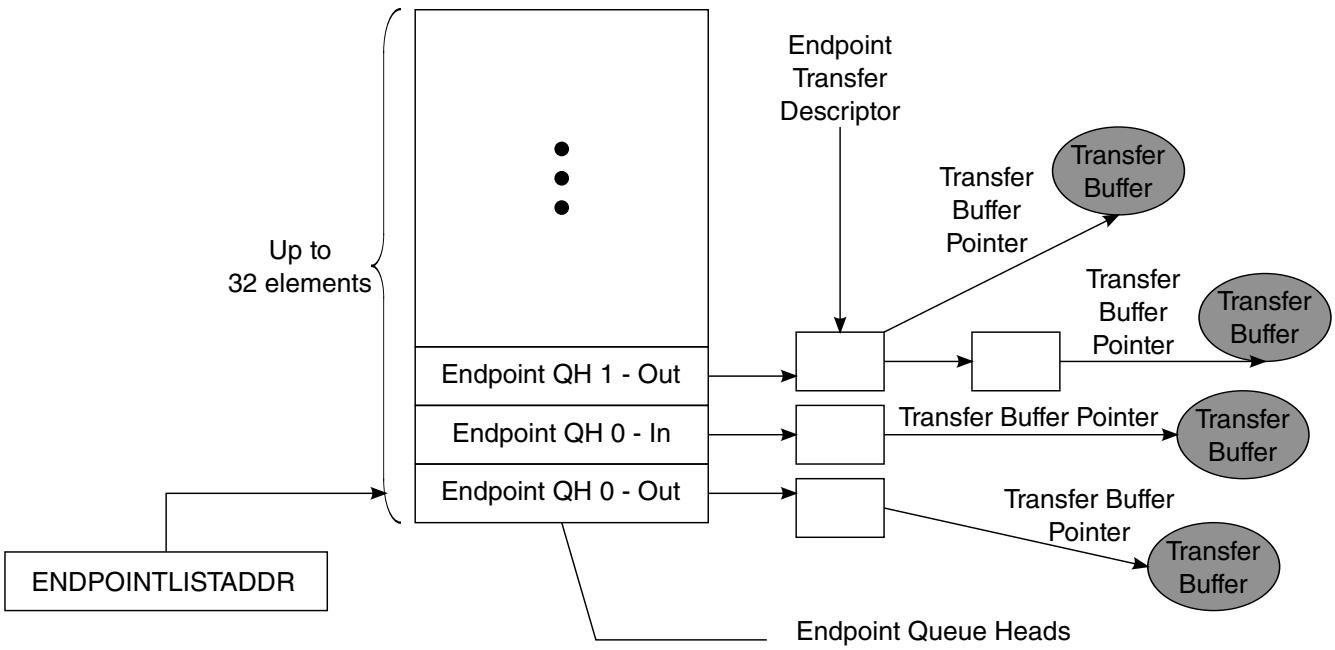


Figure 65-28. EndPoint Queue Head Organization

Endpoint queue heads are arranged in an array in a continuous area of memory pointed to by the USB.ENDPOINTLISTADDR pointer. The even -numbered device queue heads in the list support receive endpoints (OUT/SETUP) and the odd-numbered queue heads in the list are used for transmit endpoints (IN/INTERRUPT). The device controller will index into this array based upon the endpoint number received from the USB bus. All information necessary to respond to transactions for all primed transfers is contained in this list so the Device Controller can readily respond to incoming requests without having to traverse a linked list.

NOTE

The Endpoint Queue Head List must be aligned to a 2k boundary.

65.4.5.1 Endpoint Queue Head (dQH)

The device Endpoint Queue Head (dQH) is where all transfers for a given endpoint are managed. The dQH is a 48-byte data structure, but must be aligned on 64-byte boundaries.

During priming of an endpoint, the dTD (device transfer descriptor) is copied into the overlay area of the dQH, which starts at the nextTD pointer DWord and continues through the end of the buffer pointers DWords. After a transfer is complete, the dTD status DWord is updated in the dTD pointed to by the currentTD pointer. While a packet is in progress, the overlay area of the dQH is used as a staging area for the dTD so that the Device Controller can access needed information with little minimal latency.

Table 65-56. Endpoint Queue Head (dQH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mult		zlt	0		Maximum Packet Length											ios	0														
Current dTD Pointer																											0				
Next dTD Pointer																											0				T[1]
0	Total Bytes															ioc	0			MultO		0		Status							
Buffer Pointer (Page 0)																				Current Offset											
Buffer Pointer (Page 1)																				Reserved											
Buffer Pointer (Page 2)																				Reserved											
Buffer Pointer (Page 3)																				Reserved											
Buffer Pointer (Page 4)[1]																				Reserved											
Reserved																															
Set-up Buffer Bytes 3…0																															
Set-up Buffer Bytes 7…4																															

1. Transfer overlay starts at T and continues through Buffer Pointer (Page 4).

Host Controller Read/Write

Host Controller Read Only

65.4.5.1.1 Endpoint Capabilities/Characteristics

This DWord specifies static information about the endpoint, in other words, this information does not change over the lifetime of the endpoint. Device Controller software should not attempt to modify this information while the corresponding endpoint is enabled.

Table 65-57 describes the endpoint capabilities.

Table 65-57. Endpoint Capabilities/Characteristics

Bit	Description
31-30	Mult. This field is used to indicate the number of packets executed per transaction description as given by the following: 00 - Execute N Transactions as demonstrated by the USB variable length packet protocol where N is computed using the Maximum Packet Length (dQH) and the Total Bytes field (dTD) 01 Execute 1 Transaction. 10 Execute 2 Transactions. 11 Execute 3 Transactions. **NOTE:** Non-ISO endpoints must set Mult="00". ISO endpoints must set Mult="01", "10", or "11" as needed.
29	Zero Length Termination Select. This bit is used to indicate when a zero length packet is used to terminate transfers where to total transfer length is a multiple . This bit is not relevant for Isochronous 0 - Enable zero length packet to terminate transfers equal to a multiple of the Maximum Packet Length. (default). 1 - Disable the zero length packet on transfers that are equal in length to a multiple Maximum Packet Length.
28-27	Reserved. These bit reserved for future use and should be set to zero.
26-16	Maximum Packet Length. This directly corresponds to the maximum packet size of the associated endpoint (wMaxPacketSize). The maximum value this field may contain is 0x400 (1024).
15	Interrupt On Setup (IOS). This bit is used on control type endpoints to indicate if USBINT is set in response to a setup being received.
14-0	Reserved. Bits reserved for future use and should be set to zero.

65.4.5.1.2 Transfer Overlay-Endpoint Queue Head

The seven DWords in the overlay area represent a transaction working space for the device controller.

The general operational model is that the device controller can detect whether the overlay area contains a description of an active transfer. If it does not contain an active transfer, then it will not read the associated endpoint.

After an endpoint is readied, the dTD will be copied into this queue head overlay area by the device controller. Until a transfer is expired, software must not write the queue head overlay area or the associated transfer descriptor. When the transfer is complete, the device controller will write the results back to the original transfer descriptor and advance the queue.

See dTD for a description of the overlay fields.

65.4.5.1.3 Current dTD Pointer

The current dTD pointer is used by the device controller to locate the transfer in progress. This word is for Device Controller (hardware) use only and should not be modified by DCD software.

The following table describes the dTD Pointer.

Table 65-58. Next dTD Pointer

Bit	Description
31-5	Current dTD. This field is a pointer to the dTD that is represented in the transfer overlay area. This field will be modified by the Device Controller to next dTD pointer during endpoint priming or queue advance.
4-0	Reserved. Bit reserved for future use and should be set to zero.

65.4.5.1.4 Set-up Buffer

The set-up buffer is dedicated storage for the 8-byte data that follows a set-up PID.

NOTE

Each endpoint has a TX and an RX dQH associated with it, and only the RX queue head is used for receiving setup data packets.

The following table describes the Multiple Mode Control.

Table 65-59. Multiple Mode Control (HCCPARAMS)

DWord	Bits	Description
1	31-0	Setup Buffer 0. This buffer contains bytes 3 to 0 of an incoming setup buffer packet and is written by the device controller to be read by software.
2	31-0	Setup Buffer 1. This buffer contains bytes 7 to 4 of an incoming setup buffer packet and is written by the device controller to be read by software.

65.4.5.2 Endpoint Transfer Descriptor (dTD)

The dTD describes to the device controller the location and quantity of data to be sent/ received for a given transfer.

The DCD should not attempt to modify any field in an active dTD except the Next Like Pointer, which should only be modified as described in section Managing Transfers with Transfer Descriptors.

Table below shows the Endpoint Transfer Descriptor (dTD).

Table 65-60. Endpoint Transfer Descriptor (dTD)

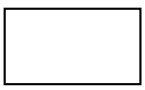
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Next Link Pointer																											0				T
0	Total Bytes															ioc	0			MultO		0		Status							
Buffer Pointer (Page 0)																				Current Offset											

Table continues on the next page...

Table 65-60. Endpoint Transfer Descriptor (dTD) (continued)

Buffer Pointer (Page 1)	0	Frame Number
Buffer Pointer (Page 2)	Reserved	
Buffer Pointer (Page 3)	Reserved	
Buffer Pointer (Page 4)	Reserved	

Host Controller Read/Write



Host Controller Read Only

The following table describes the dTD Pointer.

Table 65-61. Next dTD Pointer

Bit	Description
31-5	Next Transfer Element Pointer. This field contains the physical memory address of the next dTD to be processed. The field corresponds to memory address signals [31:5], respectively.
4-1	Reserved. Bits reserved for future use and should be set to zero.
0	Terminate (T). 1=pointer is invalid. 0=Pointer is valid (points to a valid Transfer Element Descriptor). This bit indicates to the Device Controller that there are no more valid entries in the queue.

The following table describes the dTD Token.

Table 65-62. dTD Token

Bit	Description
31	Reserved. Bit reserved for future use and should be set to zero.
30-16	Total Bytes. This field specifies the total number of bytes to be moved with this transfer descriptor. This field is decremented by the number of bytes actually moved during the transaction and only on the successful completion of the transaction. The maximum value software may store in the field is 5*4K(5000H). This is the maximum number of bytes 5 page pointers can access. Although it is possible to create a transfer up to 20K this assumes the 1st offset into the first page is 0. When the offset cannot be predetermined, crossing past the 5th page can be guaranteed by limiting the total bytes to 16K**. Therefore, the maximum recommended transfer is 16K(4000H). If the value of the field is zero when the host controller fetches this transfer descriptor (and the active bit is set), the device controller executes a zero-length transaction and retires the transfer descriptor. It is not a requirement for IN transfers that Total Bytes To Transfer be an even multiple of *Maximum Packet Length*. If software builds such a transfer descriptor for an IN transfer, the last transaction will always be less that *Maximum Packet Length*.
15	Interrupt On Complete (IOC). This bit is used to indicate if USBINT is to be set in response to device controller being finished with this dTD.
14-12	Reserved. Bits reserved for future use and should be set to zero.
11-10	Multiplier Override (MultO). This field can be used for transmit ISO's (ie. ISO-IN) to override the multiplier in the QH. This field must be zero for all packet types that are not transmit-ISO. Example:

Table continues on the next page...

Table 65-62. dTD Token (continued)

	if QH.multiplier = 3; Maximum packet size = 8; Total Bytes = 15; MultiO = 0 [default] Three packets are sent: {Data2(8); Data1(7); Data0(0)} if QH.multiplier = 3; Maximum packet size = 8; Total Bytes = 15; MultiO = 2 Two packets are sent: {Data1(8); Data0(7)} For maximal efficiency, software should compute MultO = greatest integer of (Total Bytes / Max. Packet Size) except for the case when Total Bytes = 0; then MultO should be 1. Note: Non-ISO and Non-TX endpoints must set MultO="00".
9-8	Reserved. Bits reserved for future use and should be set to zero.
7-0	Status. This field is used by the Device Controller to communicate individual command execution states back to the Device Controller software. This field contains the status of the last transaction performed on this qTD. The bit encodings are: Bit Status Field Description 7 Active. 6 Halted. 5 Data Buffer Error. 3 Transaction Error. 4,2,0 Reserved.

The table below describes the dTD Buffer Page Pointer List.

Table 65-63. dTD Buffer Page Pointer List

Bit	Description
31-12	Buffer Pointer. Selects the page offset in memory for the packet buffer. Non virtual memory systems will typically set the buffer pointers to a series of incrementing integers.
0,11-0	Current Offset. Offset into the 4kb buffer where the packet is to begin.
1,10-0	Frame Number. Written by the device controller to indicate the frame number in which a packet finishes. This is typically be used to correlate relative completion times of packets on an ISO endpoint.

65.4.6 Device Operational Model

The function of the device operation is to transfer a request in the memory image to and from the Universal Serial Bus.

Using a set of linked list transfer descriptors, pointed to by a queue head, the device controller will perform the data transfers. The following sections explain the use of the device controller from the device controller driver (DCD) point-of-view and further describe how specific USB bus events relate to status changes in the device controller programmer's interface.

65.4.6.1 Device Controller Initialization

After hardware reset, the device is disabled until the Run/Stop bit is set to a '1'. In the disabled state, the pull-up on the USB D+ is not active which prevents an attach event from occurring. At a minimum, it is necessary to have the queue heads setup for endpoint zero before the device attach occurs.

Shortly after the device is enabled, a USB reset will occur followed by setup packet arriving at endpoint 0. A Queue head must be prepared so that the device controller can store the incoming setup packet.

In order to initialize a device, the software should perform the following steps:

- Set Controller Mode in the USB.USBMODE register to device mode.

NOTE

Transitioning from host mode to device mode requires a device controller reset before modifying USB.USBMODE.

- Allocate and Initialize device queue heads in system memory.
  - Minimum: Initialize device queue heads 0 Tx & 0 Rx.

NOTE

All device queue heads for control endpoints must be initialized before the endpoint is enabled. Non-Control device queue heads before the endpoint can be used.

  - For information on device queue heads, refer to section Device Data Structures.
- Configure USB.ENDPOINTLISTADDR Pointer.
  - For additional information on USB.ENDPOINTLISTADDR, refer to the register table.
- Enable the microprocessor interrupt associated with the USB core.
  - Recommended: enable all device interrupts including: USBINT, USBERRINT, Port Change Detect, USB Reset Received, DCSuspend.
  - For a list of available interrupts refer to the Interrupt Enable Register (USBC_*n*_USBINTR) and the USB Status Register (USBC_*n*_USBSTS) register tables.
- Set Run/Stop bit to Run Mode.
  - After the Run bit is set and the device is connected to a host, a Bus Reset will be issued by host downstream port. The DCD must monitor the reset event and adjust the software state as described in the Bus Reset section of the Port State and Control section below.

NOTE

Endpoint 0 is designed as a control endpoint only and does not need to be configured using ENDPTCTRL0 register.

It is also not necessary to prime Endpoint 0 initially because the first packet received will always be a setup packet. The contents of the first setup packet will require a response in accordance with USB device framework (Chapter 9) command set.

65.4.6.2 Port State and Control

From a chip or system reset, the device controller enters the *powered* state. A transition from the *powered* state to the *attach* state occurs when the Run/Stop bit is set to a '1'.

After receiving a reset on the bus, the port will enter the *defaultFS or defaultHS* state in accordance with the reset protocol described in Appendix C.2 of the USB Specification Rev. 2.0.

The following state diagram depicts the state of a USB 2.0 device.

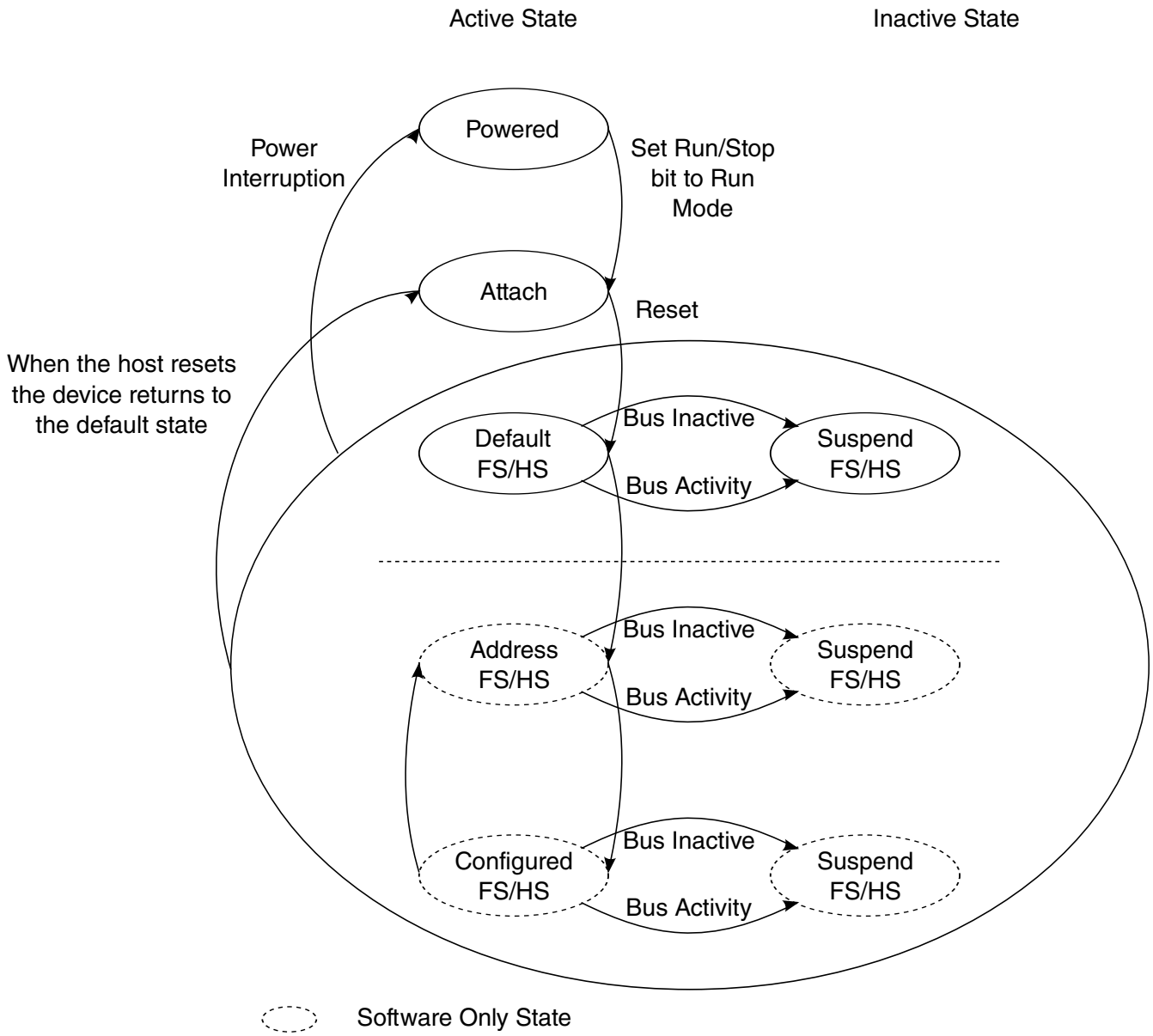


Figure 65-29. Device State Diagram

States *powered, attach, defaultFS/HS, suspendFS/HS* are implemented in the device controller and are communicated to the DCD using the following status bits:

The following table describes the Device Controller State Information Bits.

Table 65-64. Device Controller State Information Bits

Bit	Register
DCSuspend	USB Status Register (USBC_*n*_USBSTS)
USB Reset Received	USB Status Register (USBC_*n*_USBSTS)
Port Change Detect	USB Status Register (USBC_*n*_USBSTS)
High-Speed Port	Port Status & Control (USBC_*n*_PORTSC1)

It is the responsibility of the DCD to maintain a state variable to differentiate between the *DefaultFS/HS* state and the *Address/Configured* states. Change of state from *Default* to *Address* and the *Configured* states is part of the enumeration process described in the device framework section of the USB 2.0 Specification.

As a result of entering the *Address* state, the device address register (DEVICEADDR) must be programmed by the DCD.

Entry into the *Configured* indicates that all endpoints to be used in the operation of the device have been properly initialized by programming the USB_UOG_ENDPTCTRLx registers and initializing the associated queue heads.

65.4.6.2.1 Bus Reset

A bus reset is used by the host to initialize downstream devices.

When a bus reset is detected, the device controller will renegotiate its attachment speed, reset the device address to 0, and notify the DCD by interrupt (assuming the USB Reset Interrupt Enable is set). After a reset is received, all endpoints (except endpoint 0) are disabled and any primed transactions will be cancelled by the device controller. The concept of priming will be clarified below, but the DCD must perform the following tasks when a reset is received:

Clear all setup token semaphores by reading the Endpoint Status (USBC_*n*_ENDPTSTAT) register and writing the same value back to the Endpoint Status (USBC_*n*_ENDPTSTAT) register.

Clear all the endpoint complete status bits by reading the Endpoint Complete (USBC_*n*_ENDPTCOMPLETE) register and writing the same value back to the Endpoint Complete (USBC_*n*_ENDPTCOMPLETE) register.

Cancel all primed status by waiting until all bits in the Endpoint Prime (USBC_*n*_ENDPTPRIME) are 0 and then writing 0xFFFFFFFF to Endpoint Flush (USBC_*n*_ENDPTFLUSH).

Read the reset bit in the Port Status & Control (USBC_*n*_PORTSC1) register and make sure that it is still active. A USB reset will occur for a minimum of 3 ms and the DCD must reach this point in the reset cleanup before end of the reset occurs, otherwise a hardware reset of the device controller is recommended (rare.)

- A hardware reset can be performed by writing a one to the device controller reset bit in the USBCMD reset. Note: a hardware reset will cause the device to detach from the bus by clearing the Run/Stop bit. Thus, the DCD must completely re-initialize the device controller after a hardware reset.

Free all allocated dTDs because they will no longer be executed by the device controller. If this is the first time the DCD is processing a USB reset event, then it is likely that no dTDs have been allocated.

At this time, the DCD may release control back to the OS because no further changes to the device controller are permitted until a Port Change Detect is indicated.

After a Port Change Detect, the device has reached the default state and the DCD can read the Port Status & Control (USBC_*n*_PORTSC1) to determine if the device is operating in FS or HS mode. At this time, the device controller has reached normal operating mode and DCD can begin enumeration according to the USB Chapter 9 - Device Framework.

NOTE

The device DCD may use the FS/HS mode information to determine the bandwidth mode of the device

In some applications, it may not be possible to enable one or more pipes while in FS mode. *Beyond the data rate issue, there is no difference in DCD operation between FS and HS modes.*

65.4.6.2.2 Suspend/Resume

65.4.6.2.2.1 Suspend

Suspend Description

In order to conserve power, USB devices automatically enter the suspended state when the device has observed no bus traffic for a specified period. When suspended, the USB device maintains any internal status, including its address and configuration. Attached devices must be prepared to suspend at any time they are powered, regardless of if they have been assigned a non-default address, are configured, or neither. Bus activity may cease due to the host entering a suspend mode of its own. In addition, a USB device shall also enter the suspended state when the hub port it is attached to is disabled.

A USB device exits suspend mode when there is bus activity. A USB device may also request the host to exit suspend mode or selective suspend by using electrical signaling to indicate remote wakeup. The ability of a device to signal remote wakeup is optional. If the USB device is capable of remote wakeup signaling, the device must support the ability of the host to enable and disable this capability. When the device is reset, remote wakeup signaling must be disabled.

Suspend Operational Model

The device controller moves into the suspend state when suspend signaling is detected or activity is missing on the upstream port for more than a specific period. After the device controller enters the suspend state, the DCD is notified by an interrupt (assuming *DC Suspend Interrupt* is enabled). When the *DCSuspend* bit in the Port Status & Control (USBC_*n*_PORTSC1) is set to a '1', the device controller is suspended.

DCD response when the device controller is suspended is application specific and may involve switching to low power operation.

Information on the bus power limits in suspend state can be found in USB 2.0 specification.

NOTE

Review system level clocking issues defined in section (Ref: Signals-Clocking) for the clocking requirements of a suspended device controller.

65.4.6.2.2.2 Resume

If the device controller is suspended, its operation is resumed when any non-idle signaling is received on its upstream facing port. In addition, the device can signal the system to resume operation by forcing resume signaling to the upstream port.

Resume signaling is sent upstream by writing a '1' to the Resume bit in the in the Port Status & Control (USBC_*n*_PORTSC1) while the device is in suspend state. Sending resume signal to an upstream port should cause the host to issue resume signaling and bring the suspended bus segment (one more devices) back to the active condition.

NOTE

Before resume signaling can be used, the host must enable it by using the Set Feature command defined in device framework (chapter 9) of the USB 2.0 Specification.

65.4.6.2.3 Managing Endpoints

The USB 2.0 specification defines an endpoint, also called a device endpoint or an address endpoint as a uniquely addressable portion of a USB device that can source or sink data in a communications channel between the host and the device.

The endpoint address is specified by the combination of the endpoint number and the endpoint direction.

The channel between the host and an endpoint at a specific device represents a data pipe. Endpoint 0 for a device is always a *control* type data channel used for device discovery and enumeration. Other types of endpoints support by USB include *bulk, interrupt*, and

isochronous. Each endpoint type has specific behavior related to packet response and error handling. More detail on endpoint operation can be found in the USB 2.0 specification.

The USB OTG device controller hardware supports up to 8 endpoint numbers.

Each endpoint direction is essentially independent and can be configured with differing behavior in each direction. For example, the DCD can configure endpoint 1-IN to be a bulk endpoint and endpoint 1-OUT to be an isochronous endpoint. This helps to conserve the total number of endpoints required for device operation. The only exception is that control endpoints must use both directions on a single endpoint number to function as a control endpoint. Endpoint 0 is, for example, is always a control endpoint and uses the pair of directions.

Each endpoint direction requires a *queue head* allocated in memory. To support the 8 endpoint numbers, 16 *queue heads* are required. The operation of an endpoint and use of *queue heads* are described later in this document.

65.4.6.2.4 Endpoint Initialization

After hardware reset, all endpoints except endpoint zero are uninitialized and disabled. The DCD must configure and enable each endpoint by writing to configuration bit in the USB_UOG_ENDPTCTRLx register.

Each 32-bit USB_UOG_ENDPTCTRLx is split into an upper and lower half. The lower half of USB_UOG_ENDPTCTRLx is used to configure the receive or OUT endpoint and the upper half is likewise used to configure the corresponding transmit or IN endpoint. Control endpoints must be configured the same in both the upper and lower half of the USB_UOG_ENDPTCTRLx register otherwise the behavior is undefined. The following table shows how to construct a configuration word for endpoint initialization. The following table shows the fields and values for the Device Controller Endpoint initialization.

Table 65-65. Device Controller Endpoint Initialization

Field	Value
Data Toggle Reset	1
Data Toggle Inhibit	0
Endpoint Type	00 Control 01 Isochronous 10 Bulk 11 Interrupt
Endpoint Stall	0

65.4.6.2.5 Stalling

There are two occasions where the device controller may need to return to the host a STALL.

The first occasion is the functional stall, which is a condition set by the DCD as described in the USB 2.0 device framework. A functional stall is only used on non-control endpoints and can be enabled in the device controller by setting the endpoint stall bit in the USB_UOG_ENDPTCTRLx register associated with the given endpoint and the given direction. In a functional stall condition, the device controller will continue to return STALL responses to all transactions occurring on the respective endpoint and direction until the endpoint stall bit is cleared by the DCD.

A protocol stall, unlike a function stall, is used on control endpoints is automatically cleared by the device controller at the start of a new control transaction (setup phase). When enabling a protocol stall, the DCD should enable the stall bits (both directions) as a pair. A single write to the USB_UOG_ENDPTCTRLx register can ensure that both stall bits are set at the same instant.

NOTE

Any write to the USB_UOG_ENDPTCTRLx register during operational mode must preserve the endpoint type field (that is, perform a read-modify-write).

The following table shows the response matrix for the Device Controller Stall.

Table 65-66. Device Controller Stall Response Matrix

USB Packet	Endpoint Stall Bit.	Effect on STALL bit.	USB Response
SETUP packet received by a non-control endpoint.	N/A	None.	STALL
IN/OUT/PING packet received by a non-control endpoint.	'1'	None.	STALL
IN/OUT/PING packet received by a non-control endpoint.	'0'	None.	ACK/ NAK/ NYET
SETUP packet received by a control endpoint.	N/A	Cleared	ACK
IN/OUT/PING packet received by a control endpoint	'1'	None	STALL
IN/OUT/PING packet received by a control endpoint.	'0'	None.	ACK/ NAK/ NYET

65.4.6.2.6 Data Toggle

Data toggle is a mechanism to maintain data coherency between host and device for any given data pipe.

For more information on data toggle, refer to the USB 2.0 specification.

65.4.6.2.6.1 Data Toggle Reset

The DCD may reset the data toggle state bit and cause the data toggle sequence to reset in the device controller by writing a '1' to the data toggle reset bit in the USB_UOG_ENDPTCTRLx register.

This should only be necessary when configuring/initializing an endpoint or returning from a STALL condition.

65.4.6.2.6.2 Data Toggle Inhibit

NOTE

This feature is for test purposes only and should never be used during normal device controller operation.

Setting the *data toggle Inhibit bit* active ('1') causes the device controller to ignore the data toggle pattern that is normally sent and accept all incoming data packets regardless of the data toggle state.

In normal operation, the device controller checks the DATA0/DATA1 bit against the data toggle to determine if the packet is valid. If Data PID does not match the data toggle state bit maintained by the device controller for that endpoint, the Data toggle is considered not valid. If the data toggle is not valid, the device controller assumes the packet was already received and discards the packet (not reporting it to the DCD). To prevent the host controller from re-sending the same packet, the device controller will respond to the error packet by acknowledging it with either an ACK or NYET response.

65.4.6.2.6.3 Priming Transmit Endpoints

Priming a transmit endpoint will cause the device controller to fetch the device transfer descriptor (dTD) for the transaction pointed to by the device queue head (dQH).

After the dTD is fetched, it will be stored in the dQH until the device controller completes the transfer described by the dTD. Storing the dTD in the dQH allows the device controller to fetch the operating context needed to handle a request from the host without the need to follow the linked list, starting at the dQH when the host request is received.

After the device has loaded the dTD, the leading data in the packet is stored in a FIFO in the device controller. This FIFO is split into virtual channels so that the leading data can be stored for any endpoint up to the maximum number of endpoints configured at device synthesis time.

After a priming request is complete, an endpoint state of primed is indicated in the USB_UOG_ENDPTSTATUS register. For a primed transmit endpoint, the device controller can respond to an IN request from the host and meet the stringent bus turnaround time of High Speed USB.

Because only the leading data is stored in the device controller FIFO, it is necessary for the device controller to begin filling in behind leading data after the transaction starts. The FIFO must be sized to account for the maximum latency that can be incurred by the system memory bus. More information about FIFO sizing is presented in section .

65.4.6.2.6.4 Priming Receive Endpoints

Priming receive endpoints is identical to priming of transmit endpoints from the point of view of the DCD. At the device controller the major difference in the operational model is that there is no data movement of the leading packet data simply because the data is to be received from the host.

Note as part of the architecture, the FIFO for the receive endpoints is not partitioned into multiple channels like the transmit FIFO. Thus, the size of the RX FIFO does not scale with the number of endpoints.

65.4.6.3 Operational Model For Packet Transfers

All transactions on the USB bus are initiated by the host and in turn, the device must respond to any request from the host within the turnaround time stated in the USB 2.0 Specification.

At USB 1.1 Full or Low Speed rates, this turnaround time was significant and the USB 1.1 device controllers were architected so that the device controller could access main memory or interrupt a host protocol processor in order to respond to the USB 1.1 transaction. The architecture of the USB 2.0 device controller must be different because same methods will not meet USB 2.0 High-speed turnaround time requirements by simply increasing clock rate.

A USB host will send requests to the device controller in an order that can not be precisely predicted as a single pipeline, so it is not possible to prepare a single packet for the device controller to execute. However, the order of packet requests is predictable when the endpoint number and direction is considered. For example, if endpoint 3

(transmit direction) is configured as a bulk pipe, then we can expect the host will send IN requests to that endpoint. This device controller is architected in such a way that it can prepare packets for each endpoint/direction in anticipation of the host request. The process of preparing the device controller to send or receive data in response to host initiated transaction on the bus is referred to as "priming" the endpoint. This term will be used throughout the following documentation to describe the device controller operation so the DCD can be architected properly use priming. Further, note that the term "flushing" is used to describe the action of clearing a packet that was queued for execution.

65.4.6.3.1 Interrupt/Bulk Endpoint Operational Model

The behaviors of the device controller for interrupt and bulk endpoints are identical.

All valid IN and OUT transactions to bulk pipes will handshake with a NAK unless the endpoint had been primed. Once the endpoint has been primed, data delivery will commence.

A dTD will be retired by the device controller when the packets described in the transfer descriptor have been completed. Each dTD describes N packets to be transferred according to the USB Variable Length transfer protocol. The formula and table on the following page describe how the device controller computes the number and length of the packets to be sent/received by the USB vary according to the total number of bytes and maximum packet length.

With Zero Length Termination (ZLT) = 0

N = INT(Number Of Bytes/Max. Packet Length) + 1

With Zero Length Termination (ZLT) = 1

N = MAXINT(Number Of Bytes/Max. Packet Length)

Table 65-67. Variable Length Transfer Protocol Example (ZLT = 0)

Bytes (dTD)	Max. Packet Length (dQH)	N	P1	P2	P3
511	256	2	256	255	
512	256	3	256	256	0
512	512	2	512	0	

Table 65-68. Variable Length Transfer Protocol Example (ZLT = 1)

Bytes (dTD)	Max. Packet Length (dQH)	N	P1	P2	P3
511	256	2	256	255	
512	256	2	256	256	
512	512	1	512		

NOTE

The MULT field in the dQH must be set to "00" for bulk, interrupt, and control endpoints.

TX-dTD is complete when:

- All packets described dTD were successfully transmitted. *** Total bytes in dTD will equal zero when this occurs.

RX-dTD is complete when:

- All packets described in dTD were successfully received. *** Total bytes in dTD will equal zero when this occurs.
- A short packet (number of bytes < maximum packet length) was received. *** This is a successful transfer completion; DCD must check Total Bytes in dTD to determine the number of bytes that are remaining. From the total bytes remaining in the dTD, the DCD can compute the actual bytes received.
- A long packet was received (number of bytes > maximum packet size) OR (total bytes received > total bytes specified). *** This is an error condition. The device controller will discard the remaining packet, and set the Buffer Error bit in the dTD. In addition, the endpoint will be flushed and the USBERR interrupt will become active.

On the successful completion of the packet(s) described by the dTD, the active bit in the dTD will be cleared and the next pointer will be followed when the Terminate bit is clear. When the Terminate bit is set, the device controller will flush the endpoint/direction and cease operations for that endpoint/direction.

On the unsuccessful completion of a packet (see long packet above), the dQH will be left pointing to the dTD that was in error. In order to recover from this error condition, the DCD must properly reinitialize the dQH by clearing the active bit and update the nextTD pointer before attempting to re-prime the endpoint.

NOTE

All packet level errors such as a missing handshake or CRC error will be retried automatically by the device controller.

There is no required interaction with the DCD for handling such errors.

65.4.6.3.1.1 Interrupt/Bulk Endpoint Bus Response Matrix

The table below shows the response matrix for Interrput/Bulk Endpoint Bus.

Table 65-69. Interrupt/Bulk Endpoint Bus Response Matrix

	Stall	Not Primed	Primed	Underflow	Overflow
Setup	Ignore	Ignore	Ignore	N/A	N/A
In	STALL	NAK	Transmit	BS Error	N/A
Out	STALL	NAK	Receive + NYET/ACK	N/A	NAK
Ping	STALL	NAK	ACK	N/A	N/A
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore

NOTE

BS Error = Force Bit Stuff Error

NYET/ACK - NYET unless the Transfer Descriptor has packets remaining according to the USB variable length protocol then ACK.

SYSERR - System error should never occur when the latency FIFOs are correctly sized and the DCD is responsive.

65.4.6.3.2 Control Endpoint Operation Model

65.4.6.3.2.1 Setup Phase

All requests to a control endpoint begin with a setup phase followed by an optional data phase and a required status phase. The device controller will always accept the setup phase unless the setup lockout is engaged.

The setup lockout will engage so that future setup packets are ignored. Lockout of setup packets ensures that while software is reading the setup packet stored in the queue head, that data is not written as it is being read potentially causing an invalid setup packet.

The setup lockout mechanism can be disabled and a new tripwire type semaphore will ensure that the setup packet payload is extracted from the queue head without being corrupted be an incoming setup packet. This is the preferred behavior because ignoring repeated setup packets due to long software interrupt latency would be a compliance issue.

- Disable Setup Lockout by writing 1 to Setup Lockout Mode (SLOM) in USB Device Mode (USBC_*n*_USBMODE). (once at initialization). Setup lockout is not necessary when using the tripwire as described below.

NOTE

Leaving the Setup Lockout Mode As 0 will result in pre-2.3 hardware behavior.

- After receiving an interrupt and inspecting Endpoint Setup Status (USBC_*n*_ENDPTSETUPSTAT) to determine that a setup packet was received on a particular pipe:
  a. Write 1 to clear corresponding bit Endpoint Setup Status (USBC_*n*_ENDPTSETUPSTAT).
  b. Write 1 to Setup Tripwire (SUTW) in USB Command Register (USBC_*n*_USBCMD) register.
  c. Duplicate contents of dQH.SetupBuffer into local software byte array.
  d. Read Setup TripWire (SUTW) in USB Command Register (USBC_*n*_USBCMD) register. (if set - continue; if cleared - goto 2)
  e. Write 0 to clear Setup Tripwire (SUTW) in USB Command Register (USBC_*n*_USBCMD) register.
  f. Process setup packet using local software byte array copy and execute status/ handshake phases.

NOTE

After receiving a new setup packet the status and/or handshake phases may still be pending from a previous control sequence. These should be flushed & deallocated before linking a new status and/or handshake dTD for the most recent setup packet.

65.4.6.3.2.2 Data Phase

Following the setup phase, the DCD must create a device transfer descriptor for the data phase and prime the transfer.

After priming the packet, the DCD must verify a new setup packet has not been received by reading the USB.ENDPTSETUPSTAT register immediately verifying that the prime had completed. A prime will complete when the associated bit in the Endpoint Prime (USBC_*n*_ENDPTPRIME) register is zero and the associated bit in the Endpoint Status (USBC_*n*_ENDPTSTAT) register is a one. If a prime fails, ie. The Endpoint Prime (USBC_*n*_ENDPTPRIME) bit goes to zero and the Endpoint Status (USBC_*n*_ENDPTSTAT) bit is not set, then the prime has failed. This can only be due to improper setup of the dQH, dTD or a setup arriving during the prime operation. If a new setup packet is indicated after the ENDPTPRIME bit is cleared, then the transfer descriptor can be freed and the DCD must reinterpret the setup packet.

Should a setup arrive after the data stage is primed, the device controller will automatically clear the prime status (Endpoint Status (USBC_*n*_ENDPTSTAT)) to enforce data coherency with the setup packet.

NOTE

- The MULT field in the dQH must be set to "00" for bulk, interrupt, and control endpoints.
- Error handling of data phase packets is the same as bulk packets described previously.

65.4.6.3.2.3 Status Phase

Similar to the data phase, the DCD must create a transfer descriptor (with byte length equal zero) and prime the endpoint for the status phase.

The DCD must also perform the same checks of the USB.ENDPTSETUPSTAT as described above in the data phase.

NOTE

- The MULT field in the dQH must be set to 00 for bulk, interrupt, and control endpoints.
- Error handling of data phase packets is the same as bulk packets described previously.

65.4.6.3.2.4 Control Endpoint Bus Response Matrix

Shown in the following table is the device controller response to packets on a control endpoint according to the device controller state.

The table below shows the response matrix for the Control Endpoint Bus.

Table 65-70. Control Endpoint Bus Response Matrix

Token Type	Endpoint State					Setup Lockout
	Stall	Not Primed	Primed	Underflow	Overflow	
Setup	ACK	ACK	ACK	N/A	SYSERR	
In	STALL	NAK	Transmit	BS Error	N/A	N/A
Out	STALL	NAK	Receive + NYET/ACK	N/A	NAK	N/A
Ping	STALL	NAK	ACK	N/A	N/A	N/A
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore	Ignore

BS Error = Force Bit Stuff Error

NYET/ACK - NYET unless the Transfer Descriptor has packets remaining according to the USB variable length protocol then ACK.

SYSERR - System error should never occur when the latency FIFOs are correctly sized and the DCD is responsive.

65.4.6.3.3 Isochronous Endpoint Operational Model

Isochronous endpoints are used for real-time scheduled delivery of data and their operational model is significantly different than the host throttled Bulk, Interrupt, and Control data pipes.

Real time delivery by the device controller will is accomplished by the following:

- Exactly MULT Packets per (micro)Frame are transmitted/received. Note: MULT is a two-bit field in the device Queue Head. The variable length packet protocol is not used on isochronous endpoints.
- NAK responses are not used. Instead, zero length packets and sent in response to an IN request to an unprimed endpoints. For unprimed RX endpoints, the response to an OUT transaction is to ignore the packet within the device controller.
- Prime requests always schedule the transfer described in the dTD for the next (micro)frame. If the ISO-dTD is still active after that frame, then the ISO-dTD will be held ready until executed or canceled by the DCD.

An EHCI compatible host controller uses the periodic frame list to schedule data exchanges to Isochronous endpoints. The operational model for device mode does not use such a data structure. Instead, the same dTD used for Control/Bulk/Interrupt endpoints is also used for isochronous endpoints. The difference is in the handling of the dTD.

The first difference between bulk and ISO-endpoints is that priming an ISO-endpoint is a delayed operation such that an endpoint will become primed only after a SOF is received. After the DCD writes the prime bit, the prime bit will be cleared as usual to indicate to software that the device controller completed a priming the dTD for transfer. Internal to the design, the device controller hardware masks that prime start until the next frame boundary. This behavior is hidden from the DCD but occurs so that the device controller can match the dTD to a specific (micro)frame.

Another difference with isochronous endpoints is that the transaction must wholly complete in a (micro)frame. Once an ISO transaction is started in a (micro)frame it will retire the corresponding dTD when MULT transactions occur or the device controller finds a fulfillment condition.

The transaction error bit set in the status field indicates a fulfillment error condition. When a fulfillment error occurs, the frame after the transfer failed to complete wholly, the device controller will force retire the ISO-dTD and move to the next ISO-dTD.

It is important to note that fulfillment errors are only caused due to partially completed packets. If no activity occurs to a primed ISO-dTD, the transaction will stay primed indefinitely. This means it is up to software discard transmit ISO-dTDs that pile up from a failure of the host to move the data.

Finally, the last difference with ISO packets is in the data level error handling. When a CRC error occurs on a received packet, the packet is not retried similar to bulk and control endpoints. Instead, the CRC is noted by setting the *Transaction Error* bit and the data is stored as usual for the application software to sort out.

- TX Packet Retired
  - MULT counter reaches zero.
  - Fulfillment Error [*Transaction Error* bit is set]
    - # Packets Occurred > 0 AND # Packets Occurred < MULT

NOTE

For TX-ISO, MULT Counter can be loaded with a lesser value in the dTD Multiplier Override field in hardware versions 2.3 and later. If the Multiplier Override is zero, the MULT Counter is initialized to the Multiplier in the QH.

- RX Packet Retired:
  - MULT counter reaches zero.
  - Non-MDATA Data PID is received**
    - ** Exit criteria only valid in hardware version 2.3 or later. Previous to hardware version 2.3, any PID sequence that did not match the MULT field exactly would be flagged as a transaction error due to PID mismatch or fulfillment error.
  - Overflow Error:
    - Packet received is > maximum packet length. [*Buffer Error* bit is set]
    - Packet received exceeds total bytes allocated in dTD. [*Buffer Error* bit is set]
  - Fulfillment Error [*Transaction Error* bit is set]
    - # Packets Occurred > 0 AND # Packets Occurred < MULT
  - CRC Error [*Transaction Error* bit is set

NOTE

For ISO, when a dTD is retired, the next dTD is primed for the next frame. For continuous (micro)frame to (micro)frame operation the DCD should ensure that the dTD linked-list is out ahead of the device controller by at least two (micro)frames.

65.4.6.3.3.1 Isochronous Pipe Synchronization

When it is necessary to synchronize an isochroous data pipe to the host, the (micro)frame number (USB_UOG_FRINDEX register) can be used as a marker.

To cause a packet transfer to occur at a specific (micro)frame number [N], the DCD should interrupt on SOF during frame N-1. When the USB_UOG_FRINDEX=N-1, the DCD must write the prime bit. The device controller will prime the isochronous endpoint in (micro)frame N-1 so that the device controller will execute delivery during (micro)frame N.

NOTE

Priming an endpoint towards the end of (micro)frame N-1 will not guarantee delivery in (micro)frame N. The delivery may actually occur in (micro)frame N+1 if device controller does not have enough time to complete the prime before the SOF for packet N is received.

65.4.6.3.3.2 Isochronous Endpoint Bus Response Matrix

The following table shows the response matrix for the Isochronous Endpoint Bus.

Table 65-71. Isochronous Endpoint Bus Response Matrix

	Stall	Not Primed	Primed	Underflow	Overflow
Setup	STALL	STALL	STALL	N/A	N/A
In	NULL Packet	NULL Packet	Transmit	BS Error	N/A
Out	Ignore	Ignore	Receive	N/A	Drop Packet
Ping	Ignore	Ignore	Ignore	Ignore	Ignore
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore

1. BS Error = Force Bit Stuff Error

   NULL Packet = Zero Length Packet

65.4.6.4 Managing Queue Heads

The following figure shows the End Point Queue Head.

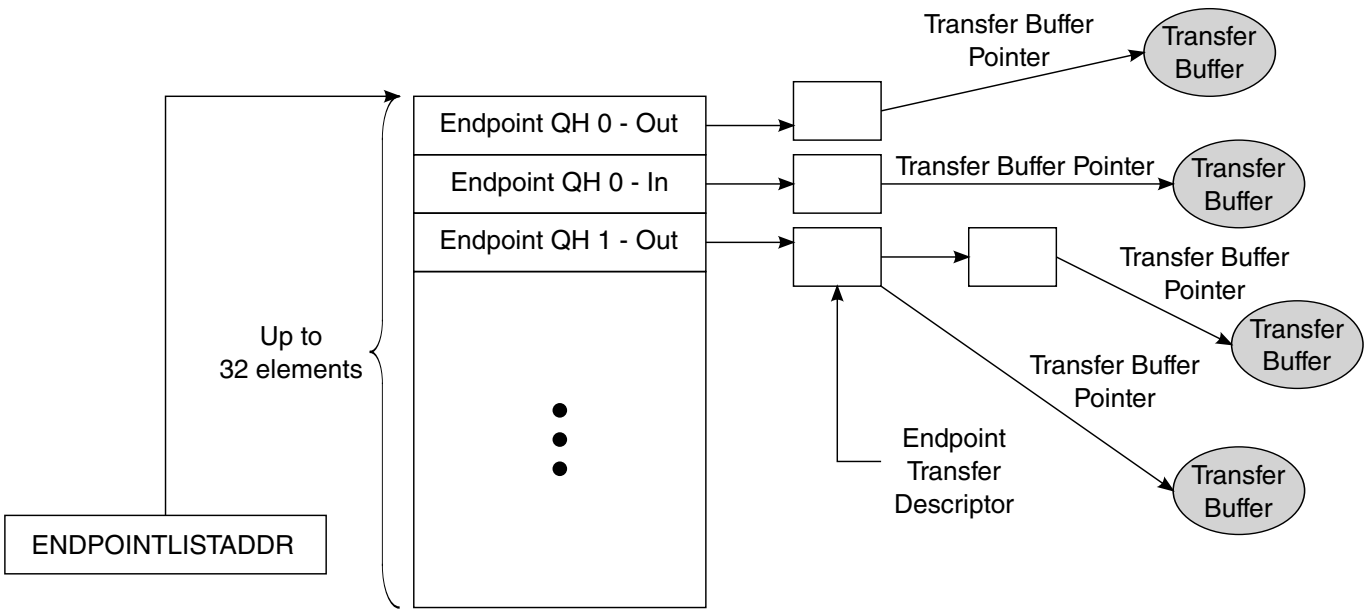


Figure 65-30. End Point Queue Head Diagram

The device queue head (dQH) points to the linked list of transfer tasks, each depicted by the device Transfer Descriptor (dTD). An area of memory pointed to by USB.ENDPOINTLISTADDR contains a group of all dQH's in a sequential list as shown in Figure 65-30. The even elements in the list of dQH's are used for receive endpoints (OUT/SETUP) and the odd elements are used for transmit endpoints (IN/INTERRUPT). Device transfer descriptors are linked head to tail starting at the queue head and ending at a terminate bit. Once the dTD has been retired, it will no longer be part of the linked list from the queue head. Therefore, software is required to track all transfer descriptors because pointers will no longer exist within the queue head once the dTD is retired (see section Software Link Pointers).

In addition to the current and next pointers and the dTD overlay examined in section Operational Model For Packet Transfers, the dQH also contains the following parameters for the associated endpoint: Multipler, Maximum Packet Length, Interrupt On Setup. The complete initialization of the dQH including these fields is demonstrated in the next section.

65.4.6.4.1 Queue Head Initialization

One device queue head must be initialized for each active endpoint.

To initialize a device queue head:

- Write the wMaxPacketSize field as required by the USB Chapter 9 or application specific protocol.
- Write the multiplier field to 0 for control, bulk, and interrupt endpoints. For ISO endpoints, set the multiplier to 1,2, or 3 as required bandwidth an in conjunction with the USB Chapter 9 protocol.

NOTE

In FS mode, the multiplier field can only be 1 for ISO endpoints.

- Write the next dTD Terminate bit field to 1.
- Write the Active bit in the status field to 0.
- Write the Halt bit in the status field to 0.

NOTE

The DCD must only modify dQH if the associated endpoint is not primed and there are no outstanding dTD's.

65.4.6.4.2 Operational Model For Setup Transfers

As discussed in section Control Endpoint Operation Model, setup transfer requires special treatment by the DCD. A setup transfer does not use a dTD but instead stores the incoming data from a setup packet in an 8-byte buffer within the dQH.

Upon receiving notification of the setup packet, the DCD should handle the setup transfer as demonstrated here:

1. Copy setup buffer contents from dQH - RX to software buffer.
2. Acknowledge setup backup by writing a "1" to the corresponding bit in ENDPTSETUPSTAT.

NOTE

- The acknowledge must occur before continuing to process the setup packet.
- After the acknowledge has occurred, the DCD must not attempt to access the setup buffer in the dQH - RX. Only the local software copy should be examined.

3. Check for pending data or status dTD's from previous control transfers and flush if any exist as discussed in section Flushing/De-priming an Endpoint.
4. Decode setup packet and prepare data phase [optional] and status phase transfer as required by the USB Chapter 9 or application specific protocol.

NOTE

It is possible for the device controller to receive setup packets before previous control transfers complete. Existing control packets in progress must be flushed and the new control packet completed.

65.4.6.5 Managing Transfers with Transfer Descriptors

65.4.6.5.1 Software Link Pointers

It is necessary for the DCD software to maintain head and tail pointers to the for the linked list of dTDs for each respective queue head.

This is necessary because the dQH only maintains pointers to the current working dTD and the next dTD to be executed. The operations described in next section for managing dTD will assume the DCD can use reference the head and tail of the dTD linked list. The following figure shows the Software Link Pointers.

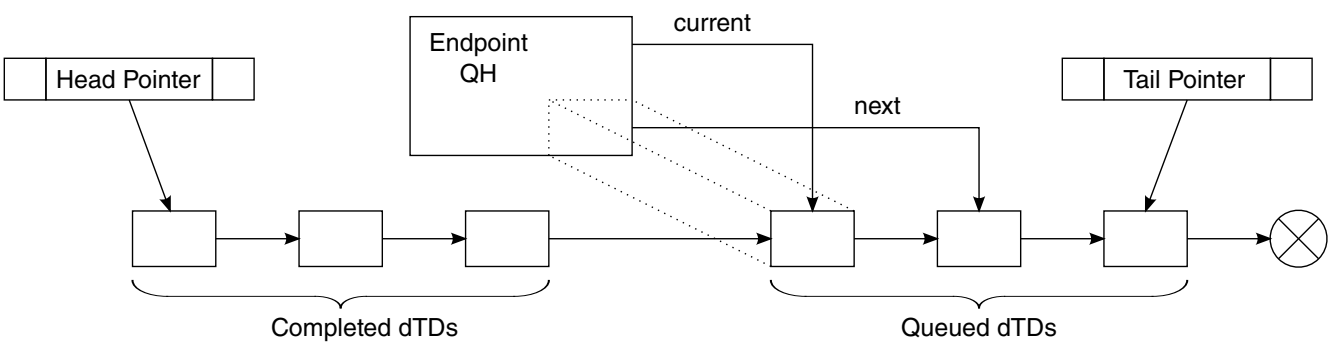


Figure 65-31. Software Link Pointers

NOTE

To conserve memory, the reserved fields at the end of the dQH can be used to store the Head & Tail pointers, but it still remains the responsibility of the DCD to maintain the pointers.

65.4.6.5.2 Building a Transfer Descriptor

Before a transfer can be executed from the linked list, a dTD must be built to describe the transfer.

Use the following procedure for building dTDs.

Allocate 8-DWord dTD block of memory aligned to 8-DWord boundaries. Example: bit address 4:0 would be equal to "00000"

Write the following fields:

1. Initialize first 7 DWords to 0.
2. Set the terminate bit to 1.
3. Fill in total bytes with transfer size.
4. Set the interrupt on complete if desired.
5. Initialize the status field with the active bit set to 1 and all remaining status bits set to 0.
6. Fill in buffer pointer page 0 and the current offset to point to the start of the data buffer.
7. Initialize buffer pointer page 1 through page 4 to be one greater than each of the previous buffer pointer.

65.4.6.5.3 Executing A Transfer Descriptor

To safely add a dTD, the DCD must be follow this procedure which will handle the event where the device controller reaches the end of the dTD list at the same time a new dTD is being added to the end of the list.

Determine whether the link list is empty: Check DCD driver to see if pipe is empty (internal representation of linked-list should indicate if any packets are outstanding).

- Case 1: Link list is empty
  a. Write dQH next pointer AND dQH terminate bit to 0 as a single DWord operation.
  b. Clear active & halt bit in dQH (in case set from a previous error).
  c. Prime endpoint by writing 1 to correct bit position in Endpoint Prime (USBC_*n*_ENDPTPRIME).
- Case 2: Link list is not empty
  a. Add dTD to end of linked list.
  b. Read correct prime bit in Endpoint Prime (USBC_*n*_ENDPTPRIME)- if 1 DONE.
  c. Set ATDTW bit in USBCMD register to 1.
  d. Read correct status bit in Endpoint Status (USBC_*n*_ENDPTSTAT). (store in tmp. variable for later)
  e. Read ATDTW bit in USBCMD register.
     - If 0 goto 3.
     - If 1 continue to 6.
  f. Write ATDTW bit in USBCMD register to 0.

g. If status bit read in (3) is 1 DONE.
h. If status bit read in (3) is 0 then Goto Case 1: Step 1.

65.4.6.5.4 Transfer Completion

After a dTD has been initialized and the associated endpoint primed the device controller will execute the transfer upon the host-initiated request. The DCD will be notified with a USB interrupt if the Interrupt On Complete bit was set or alternately, the DCD can poll the endpoint complete register to find when the dTD had been executed. After a dTD has been executed, DCD can check the status bits to determine success or failure.

NOTE

Multiple dTD can be completed in a single endpoint complete notification. After clearing the notification, DCD must search the dTD linked list and retire all dTDs that have finished (Active bit cleared).

By reading the status fields of the completed dTDs, the DCD can determine if the transfers completed successfully. Success is determined with the following combination of status bits:

- Active = 0
- Halted = 0
- Transaction Error = 0
- Data Buffer Error = 0

Should any combination other than the one shown above exist, the DCD must take proper action. Transfer failure mechanisms are indicated in the Device Error Matrix.

In addition to checking the status bit the DCD must read the Transfer Bytes field to determine the actual bytes transferred. When a transfer is complete, the Total Bytes transferred is by decremented by the actual bytes transferred. For Transmit packets, a packet is only complete after the actual bytes reaches zero, but for receive packets, the host may send fewer bytes in the transfer according the USB variable length packet protocol.

65.4.6.5.5 Flushing/De-priming an Endpoint

It is necessary for the DCD to flush to de-prime one more endpoints on a USB device reset or during a broken control transfer.

There may also be application specific requirements to stop transfers in progress. The following procedure can be used by the DCD to stop a transfer in progress:

1. Write a '1' to the corresponding bit(s) in Endpoint Flush (USBC_*n*_ENDPTFLUSH).

2. Wait until all bits in Endpoint Flush (USBC_*n*_ENDPTFLUSH) are '0'.
   - Software note: this operation may take a large amount of time depending on the USB bus activity. It is not desirable to have this wait loop within an interrupt service routine.
3. Read Endpoint Status (USBC_*n*_ENDPTSTAT) to ensure that for all endpoints commanded to be flushed, that the corresponding bits are now '0'. If the corresponding bits are '1' after step #2 has finished, then the flush failed as described in the following:
   - Explanation: In very rare cases, a packet is in progress to the particular endpoint when commanded flush using Endpoint Flush (USBC_*n*_ENDPTFLUSH). A safeguard is in place to refuse the flush to ensure that the packet in progress completes successfully. The DCD may need to repeatedly flush any endpoints that fail to flush be repeating steps 1-3 until each endpoint is successfully flushed.

65.4.6.5.6 Device Error Matrix

The following table summarizes packet errors that are not automatically handled by the Device Controller.

Table 65-72. Device Error Matrix

Error	Direction	Packet Type	Data Buffer Error Bit	Transaction Error Bit
Overflow **	RX	Any	1	0
ISO Packet Error	RX	ISO	0	1
ISO Fulfillment Error	Both	ISO	0	1

Notice that the device controller handles all errors on Bulk/Control/Interrupt Endpoints except for a data buffer overflow. However, for ISO endpoints, errors packets are not retried and errors are tagged as indicated. The table below describes the errors.

Table 65-73. Error Descriptions

Error	Description
Overflow	Number of bytes received exceeded max. packet size or total buffer length. ** This error will also set the Halt bit in the dQH and if there are dTDs remaining in the linked list for the endpoint, then those will not be executed.
ISO Packet Error	CRC Error on received ISO packet. Contents not guaranteed to be correct.
ISO Fulfillment Error	Host failed to complete the number of packets defined in the dQH mult field within the given (micro)frame. For scheduled data delivery the DCD may need to readjust the data queue because a fulfillment error will cause Device Controller to cease data transfers on the pipe for one (micro)frame. During the "dead" (micro)frame, the Device Controller reports error on the pipe and primes for the following frame.

65.4.6.6 Servicing Interrupts

The interrupt service routine must consider that there are high-frequency, low-frequency operations, and error operations and order accordingly.

65.4.6.6.1 High-Frequency Interrupts

High frequency interrupts in particular should be handled in the order below. The most important of these is listed first because the DCD must acknowledge a setup buffer in the timeliest manner possible.

The table below describes the High frequency interrupt events.

Table 65-74. High Frequency Interrupt Events

Execution Order	Interrupt	Action
1a	USB Interrupt - USB.ENDPTSETUPSTATUS	Copy contents of setup buffer and acknowledge setup packet (as indicated in Figure 65-30 shows the End Point Queue Head). Process setup packet according to USB 2.0 Chapter 9 or application specific protocol.
1b	USB Interrupt[1] - USB.ENDPTCOMPLETE	Handle completion of dTD as indicated in Figure 65-30 shows the End Point Queue Head.
2	SOF Interrupt	Action as deemed necessary by application. This interrupt may not have a use in all applications.

1. It is likely that multiple interrupts to stack up on any call to the Interrupt Service Routine AND during the Interrupt Service Routine.

65.4.6.6.2 Low-Frequency Interrupts

The low frequency interrupts can be handled in any order because they do not occur often in comparison to the high-frequency interrupts.

The table below shows the Low frequency interrupt events.

Table 65-75. Low Frequency Interrupt Events

Interrupt	Action
Port Change	Change software state information.
Sleep Enable (Suspend)	Change software state information. Low power handling as necessary.
Reset Received	Change software state information. Abort pending transfers.

65.4.6.6.3 Error Interrupts

Error interrupts will be least frequent and should be placed last in the interrupt service routine.

The following table shows the error interrput events.

Table 65-76. Error Interrupt Events

Interrupt	Action
USB Error Interrupt	This error is redundant because it combines USB Interrupt and an error status in the dTD. The DCD will more aptly handle packet-level errors by checking dTD status field upon receipt of USB Interrupt (w/ USB.ENDPTCOMPLETE).
System Error	Unrecoverable error. Immediate Reset of core; free transfers buffers in progress and restart the DCD.

65.5 USB Non-Core Memory Map/Register Definition

There are two kinds of registers in the USB module: USB core registers and USB non-core registers.

USB core registers are used to control USB core functions, and more independent of USB features. Each USB controller core has its own core registers.

USB non-core registera are additional to USB core registers, and more dependent on USB features. i.MX series products vary in non-core registers.

This section describes only the USB non-core registers. For detailed descriptions of USB core registers, please refer to Register Interface .

NOTE

For reserved bits, please preserve the value when writing (read its reset value, then write this value back).

"USB_UOG_", "USB_UH1_", "USB_UH2_", "USB_UH3_" prefix in register name indicates it is a core register for OTG/ Host1/Host2/Host3 controller core respectively.

"USB_USB_" prefix in register name indicates it is a USB non-core register.

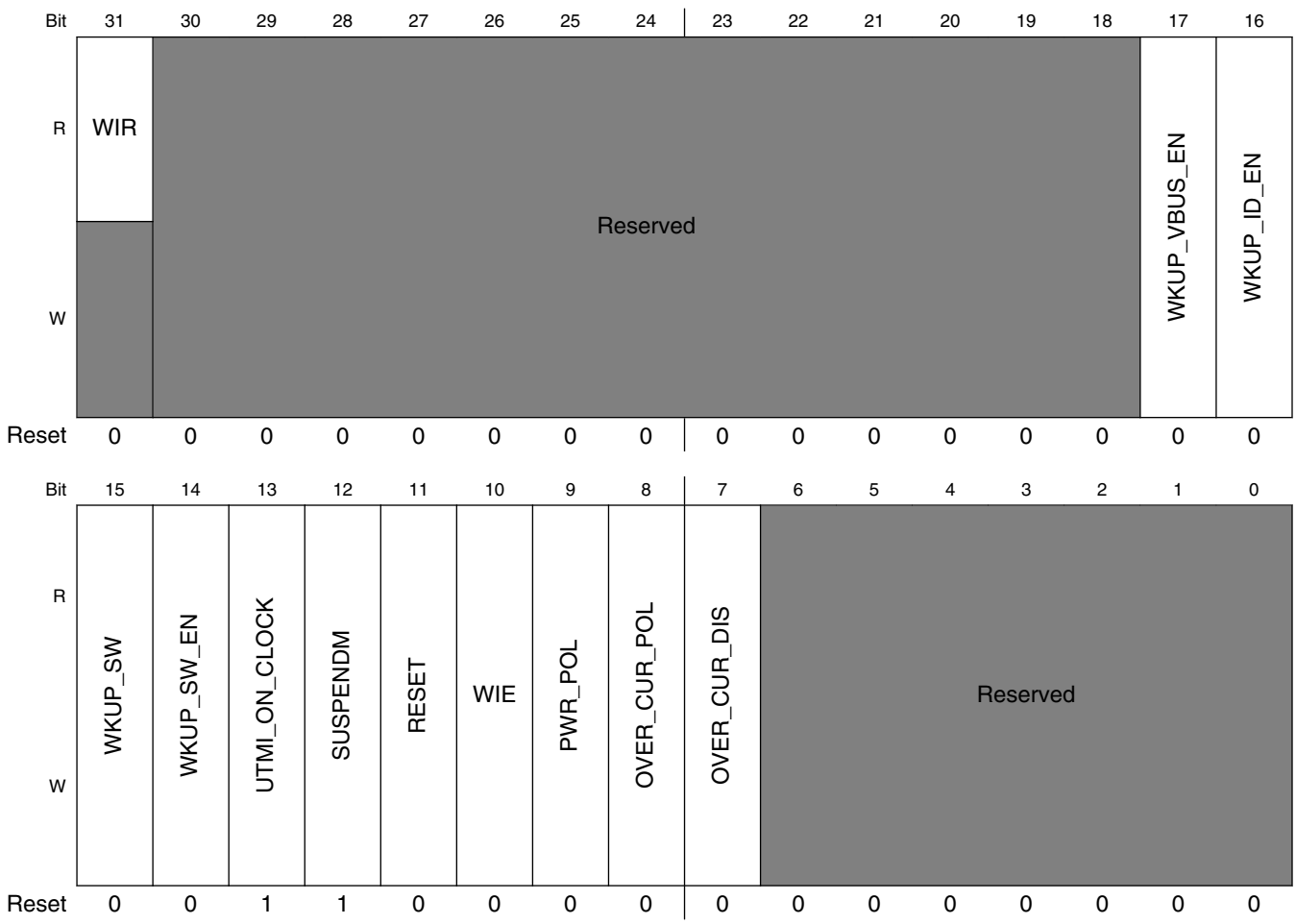
USBNC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
218_4800	USB OTG Control Register (USBNC_USB_OTG_CTRL)	32	R/W	0000_3000h	65.5.1/5424
218_4804	USB Host1 Control Register (USBNC_USB_UH1_CTRL)	32	R/W	0000_3000h	65.5.2/5427
218_4808	USB Host2 Control Register (USBNC_USB_UH2_CTRL)	32	R/W	0000_1000h	65.5.3/5430
218_480C	USB Host3 Control Register (USBNC_USB_UH3_CTRL)	32	R/W	0000_1000h	65.5.4/5432
218_4810	USB Host2 HSIC Control Register (USBNC_USB_UH2_HSIC_CTRL)	32	R/W	0000_0042h	65.5.5/5434
218_4814	USB Host3 HSIC Control Register (USBNC_USB_UH3_HSIC_CTRL)	32	R/W	0000_0042h	65.5.6/5436
218_4818	OTG UTMI PHY Control 0 Register (USBNC_USB_OTG_PHY_CTRL_0)	32	R/W	0000_0000h	65.5.7/5437
218_481C	Host1 UTMI PHY Control 0 Register (USBNC_USB_UH1_PHY_CTRL_0)	32	R/W	0000_0098h	65.5.8/5438

65.5.1 USB OTG Control Register (USBNC_USB_OTG_CTRL)

The USB OTG control register controls the integration specific features of the USB OTG module. These features are not directly related to the USB functionality, but control special features, interfacing on the USB ports, as well as power control and wake-up functionality.

Address: 218_4000h base + 800h offset = 218_4800h



USBNC_USB_OTG_CTRL field descriptions

Field	Description
31 WIR	OTG Wake-up Interrupt Request This bit indicates that a wake-up interrupt request is received on the OTG port. This bit is cleared by disabling the wake-up interrupt (clearing bit "OWIE"). 1 Wake-up Interrupt Request received 0 No wake-up interrupt request received

Table continues on the next page...

USBNC_USB_OTG_CTRL field descriptions (continued)

Field	Description
30–18 -	This field is reserved. Reserved
17 WKUP_VBUS_EN	OTG wake-up on VBUS change enable 1 Enable 0 Disable
16 WKUP_ID_EN	OTG Wake-up on ID change enable 1 Enable 0 Disable
15 WKUP_SW	OTG Software Wake-up 1 Force wake-up 0 Inactive
14 WKUP_SW_EN	OTG Software Wake-up Enable 1 Enable 0 Disable
13 UTMI_ON_CLOCK	Force OTG UTMI PHY clock output on even if suspend mode. 1 Force clock output on 0 Inactive
12 SUSPENDM	Force OTG UTMI PHY Suspend. For Freescale test only. This bit is used to put PHY into low-power suspend mode. During normal operation, S/W should set bits SUSP and PHCD in USB core register PORTSC1 to put PHY into suspend mode. 1 Inactive 0 Force OTG UTMI PHY Suspend
11 RESET	Force OTG UTMI PHY Reset This bit is used to force a reset to the UTMI PHY. For Freescale test only. During normal operation, S/W should use USBCMD.RST bit to reset the UTMI PHY 1 Reset the PHY 0 Inactive
10 WIE	OTG Wake-up Interrupt Enable This bit enables or disables the OTG wake-up interrupt. Disabling the interrupt also clears the Interrupt request bit. Wake-up interrupt enable should be turned off after receiving a wake-up interrupt and turned on again prior to going in suspend mode 1 Interrupt Enabled 0 Interrupt Disabled
9 PWR_POL	OTG Power Polarity This bit should be set according to power switch's enable polarity. 1 Power switch has an active-high enable input 0 Power switch has an active-low enable input

Table continues on the next page...

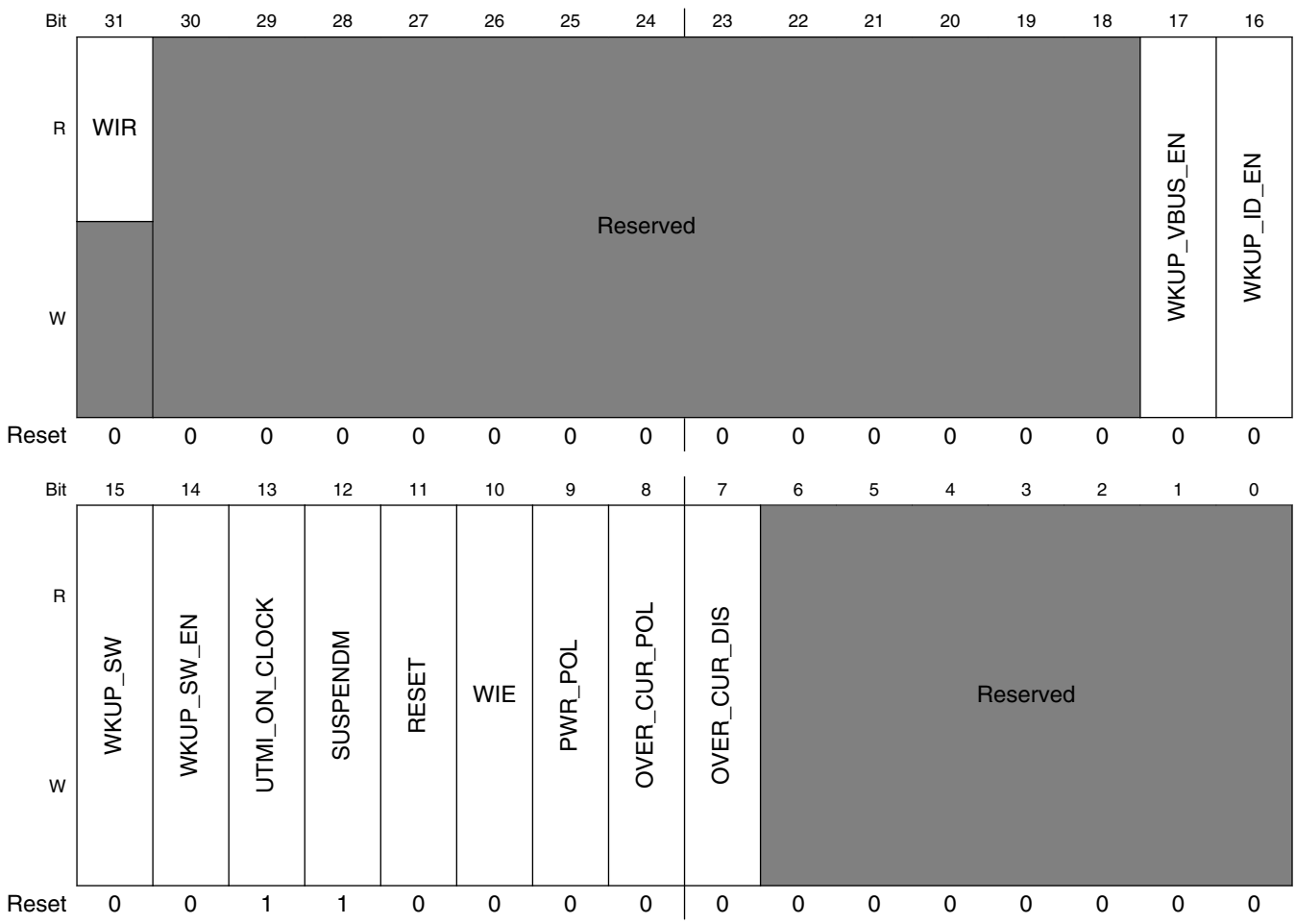
USBNC_USB_OTG_CTRL field descriptions (continued)

Field	Description
8 OVER_CUR_POL	OTG Polarity of Overcurrent The polarity of OTG port overcurrent event 1 Low active (low on this signal represents an overcurrent condition) 0 High active (high on this signal represents an overcurrent condition)
7 OVER_CUR_DIS	Disable OTG Overcurrent Detection 1 Disables overcurrent detection 0 Enables overcurrent detection
-	This field is reserved. Reserved

65.5.2 USB Host1 Control Register (USBNC_USB_UH1_CTRL)

The USB Host1 control register controls the integration specific features of the USB Host1 module. These features are not directly related to the USB functionality, but control special features, interfacing on the USB ports, as well as power control and wake-up functionality.

Address: 218_4000h base + 804h offset = 218_4804h



USBNC_USB_UH1_CTRL field descriptions

Field	Description
31 WIR	Host 1 Wake-up Interrupt Request This bit indicates that a wake-up interrupt request is received on the OTG port. This bit is cleared by disabling the wake-up interrupt (clearing bit "OWIE"). 1 Wake-up Interrupt Request received 0 No wake-up interrupt request received

Table continues on the next page...

USBNC_USB_UH1_CTRL field descriptions (continued)

Field	Description
30–18 - Reserved	This field is reserved.
17 WKUP_VBUS_EN	Host 1 wake-up on VBUS change enable 1 Enable 0 Disable
16 WKUP_ID_EN	Host 1 Wake-up on ID change enable 1 Enable 0 Disable
15 WKUP_SW	Host 1 Software Wake-up 1 Force wake-up 0 Inactive
14 WKUP_SW_EN	Host 1 Software Wake-up Enable 1 Enable 0 Disable
13 UTMI_ON_CLOCK	Force Host 1 UTMI PHY clock output on even if in low-power suspend mode. 1 Enable 0 Disable
12 SUSPENDM	Force Host 1 UTMI PHY Suspend. For Freescale test only. This bit is used to put PHY into low-power suspend mode. During normal operation, S/W should set bits SUSP and PHCD in USB core register PORTSC1 to put PHY into suspend mode. 1 Disable 0 Enable
11 RESET	Force Host 1 UTMI PHY Reset. For Freescale test only. This bit is used to force a reset to the UTMI PHY. During normal operation, S/W should use USBCMD.RST bit to reset the UTMI PHY 1 Reset the PHY 0 Inactive
10 WIE	Host 1 Wake-up Interrupt Enable This bit enables or disables the Host 1 wake-up interrupt. Disabling the interrupt also clears the Interrupt request bit. Wake-up interrupt enable should be turned off after receiving a wake-up interrupt and turned on again prior to going in suspend mode 1 Interrupt Enabled 0 Interrupt Disabled
9 PWR_POL	Host1 Power Polarity This bit should be set according to the power switch's enable polarity. 1 Power switch has an active-high enable input 0 Power switch has an active-low enable input

Table continues on the next page...

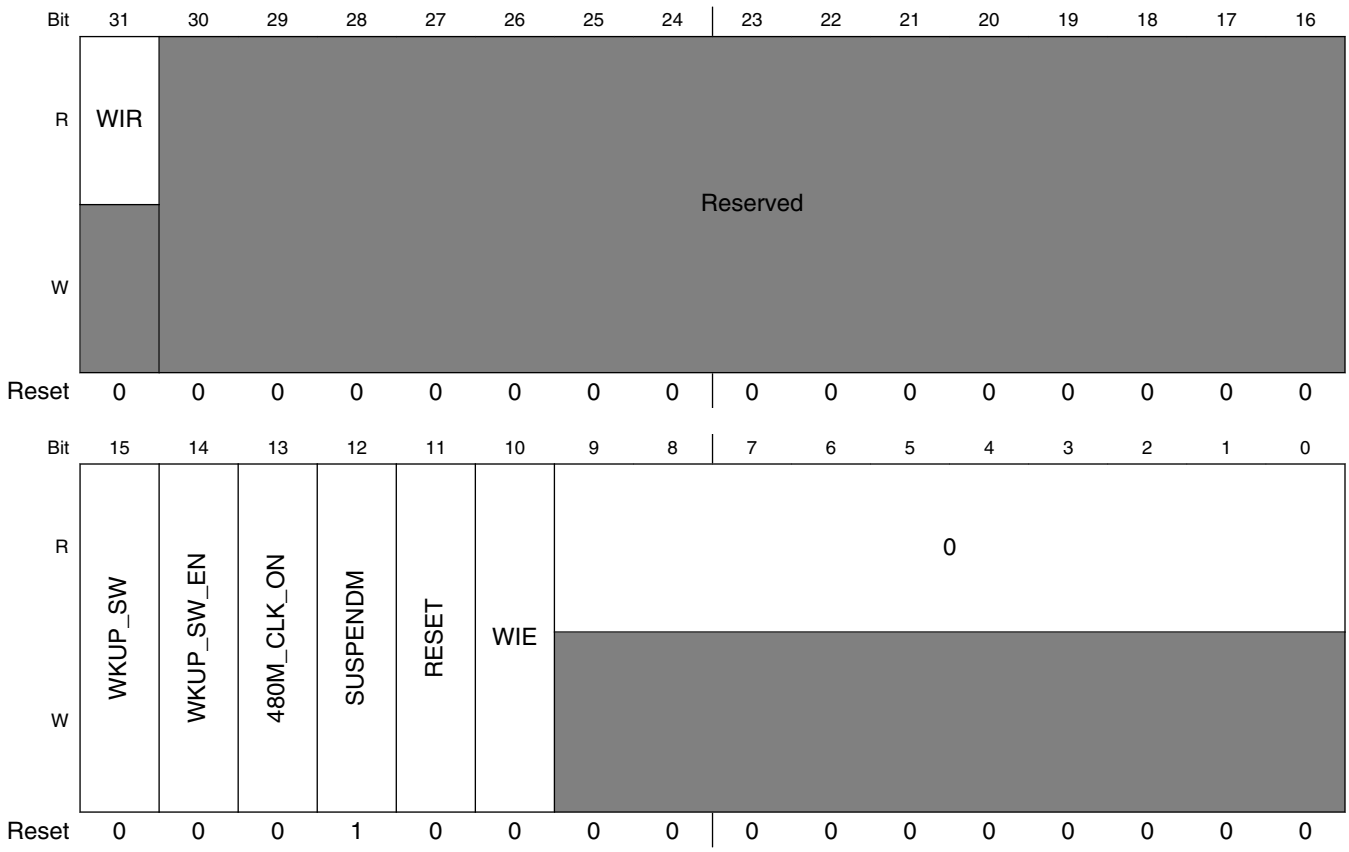
USBNC_USB_UH1_CTRL field descriptions (continued)

Field	Description
8 OVER_CUR_POL	Host 1 Polarity of Overcurrent The polarity of Host 1 port overcurrent event 1 Low active (low on this signal represents an overcurrent condition) 0 High active (high on this signal represents an overcurrent condition)
7 OVER_CUR_DIS	Disable Host 1 Overcurrent Detection 1 Disables overcurrent detection 0 Enables overcurrent detection
-	This field is reserved. Reserved

65.5.3 USB Host2 Control Register (USBNC_USB_UH2_CTRL)

The USB Host2 control register controls the integration specific features of the USB host2 module. These features are not directly related to the USB functionality, but control special features, interfacing on the USB ports, as well as power control and wake-up functionality.

Address: 218_4000h base + 808h offset = 218_4808h



USBNC_USB_UH2_CTRL field descriptions

Field	Description
31 WIR	Host 2 Wake-up Interrupt Request This bit indicates that a wake-up interrupt request is received on the Host 2 port. This bit is cleared by disabling the wake-up interrupt (clearing bit "OWIE"). 1 Wake-up Interrupt Request received 0 No wake-up interrupt request received
30–16 -	This field is reserved. Reserved

Table continues on the next page...

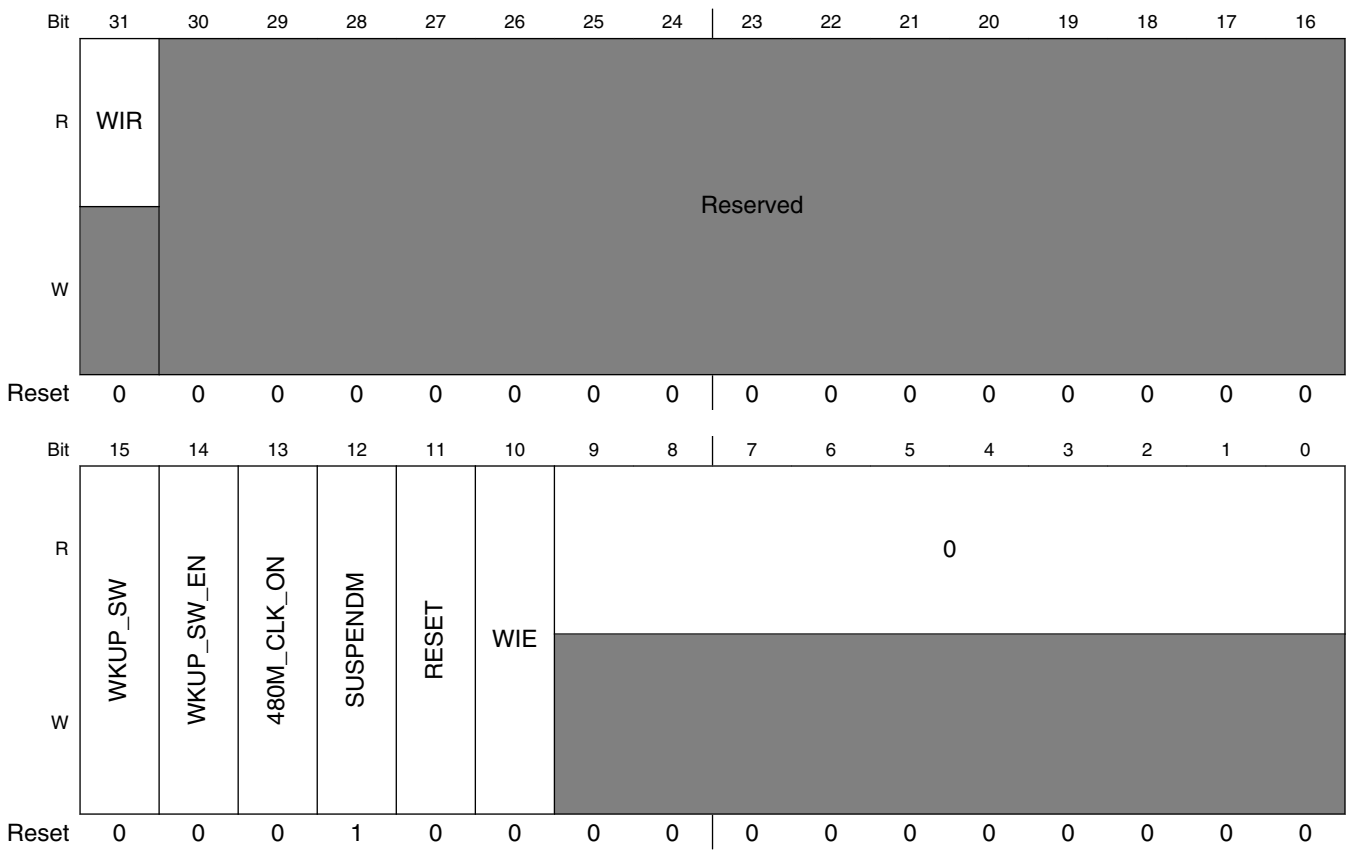
USBNC_USB_UH2_CTRL field descriptions (continued)

Field	Description
15 WKUP_SW	Host 2 Software Wake-up 1 Force wake-up 0 Inactive
14 WKUP_SW_EN	Host 2 Software Wake-up Enable 1 Enable 0 Disable
13 480M_CLK_ON	Force OTG UTMI PHY 480M clock output on when Host 2 is not in suspend mode. 1 Force OTG UTMI PHY 480M clock output on 0 Inactive
12 SUSPENDM	Force Host 2 UTMI PHY Suspend This bit is used to put PHY into suspend mode. During normal operation, S/W should set bits SUSP and PHCD in USB core register PORTSC1 to put PHY into suspend mode. For Freescale test only. 1 Disable 0 Enable
11 RESET	Force Host 2 UTMI PHY Reset This bit is used to force a reset to the UTMI PHY. During normal operation, S/W should set USBCMD.RST bit to reset the UTMI PHY For Freescale test only. 1 Reset the PHY 0 Inactive
10 WIE	Host 2 Wake-up Interrupt Enable This bit enables or disables the Host 2 wake-up interrupt. Disabling the interrupt also clears the Interrupt request bit. Wake-up interrupt enable should be turned off after receiving a wake-up interrupt and turned on again prior to going in suspend mode 1 Interrupt Enabled 0 Interrupt Disabled
Reserved	This read-only field is reserved and always has the value 0.

65.5.4 USB Host3 Control Register (USBNC_USB_UH3_CTRL)

The USB Host3 control register controls the integration specific features of the USB Host3 module. These features are not directly related to the USB functionality, but control special features, interfacing on the USB ports, as well as power control and wake-up functionality.

Address: 218_4000h base + 80Ch offset = 218_480Ch



USBNC_USB_UH3_CTRL field descriptions

Field	Description
31 WIR	Host 3 Wake-up Interrupt Request This bit indicates that a wake-up interrupt request is received on the OTG port. This bit is cleared by disabling the wake-up interrupt (clearing bit "OWIE"). 1 Wake-up interrupt received 0 No Wake-up interrupt received
30–16 -	This field is reserved. Reserved

Table continues on the next page...

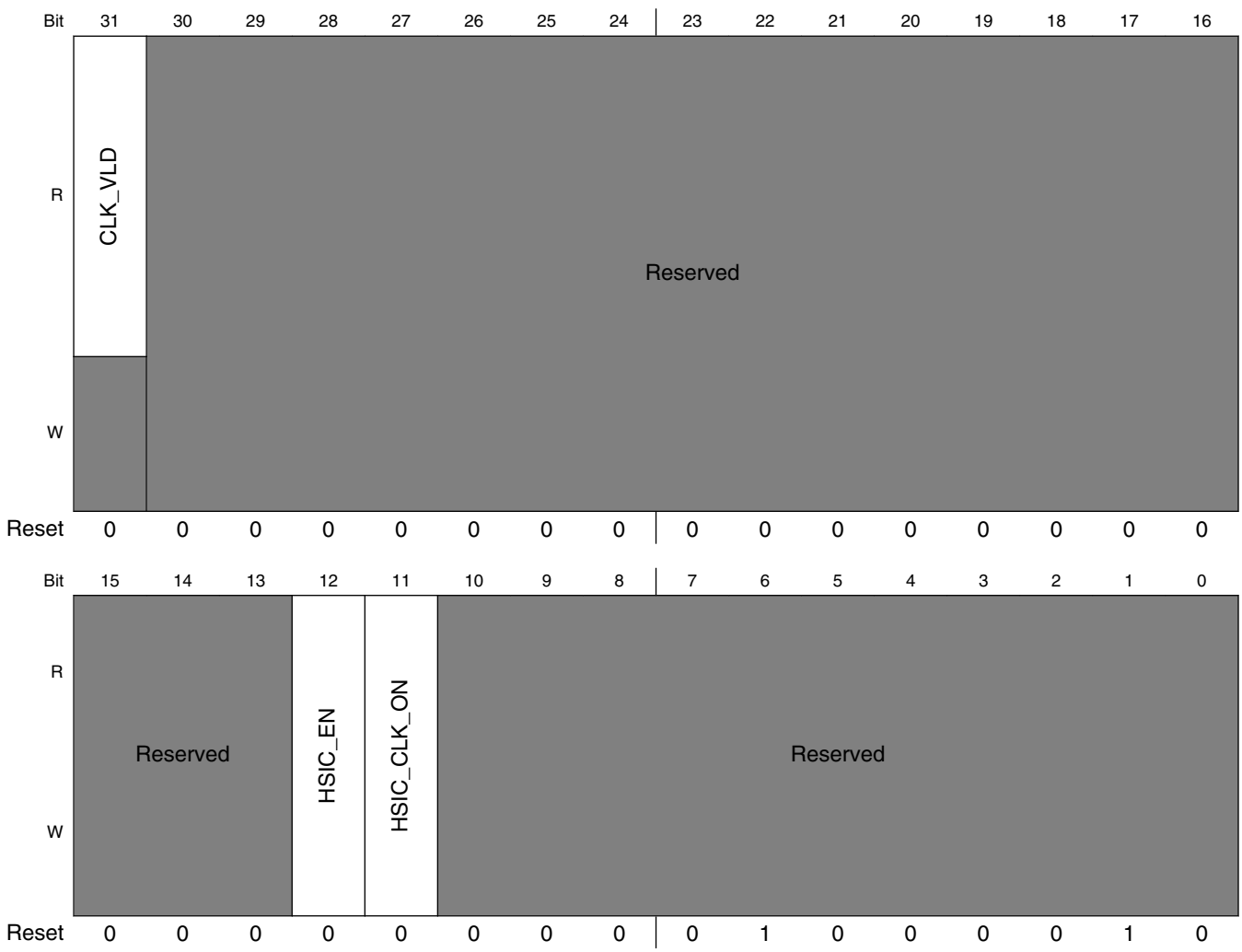
USBNC_USB_UH3_CTRL field descriptions (continued)

Field	Description
15 WKUP_SW	Host 3 Software Wake-up 1 Force wake-up 0 Inactive
14 WKUP_SW_EN	Host 3 Software Wake-up Enable 1 Enable 0 Disable
13 480M_CLK_ON	Force OTG UTMI PHY 480M clock output on when Host 3 is not in suspend mode. 1 Force OTG UTMI PHY 480M clock output on 0 Inactive
12 SUSPENDM	Force Host 3 UTMI PHY Suspend This bit is used to put PHY into suspend mode. During normal operation, S/W should set bits SUSP and PHCD in USB core register PORTSC1 to put PHY into suspend mode. For Freescale test only. 1 Inactive 0 Force OTG UTMI PHY Suspend
11 RESET	Force Host 3 UTMI PHY Reset This bit is used to force a reset to the UTMI PHY. During normal operation, S/W should set USBCMD.RST bit to reset the UTMI PHY For Freescale test only. 1 Reset the PHY 0 Inactive
10 WIE	Host 3 Wake-up Interrupt Enable This bit enables or disables the Host 3 wake-up interrupt. Disabling the interrupt also clears the Interrupt request bit. Wake-up interrupt enable should be turned off after receiving a wake-up interrupt and turned on again prior to going in suspend mode 1 Interrupt Enabled 0 Interrupt Disabled
Reserved	This read-only field is reserved and always has the value 0.

65.5.5 USB Host2 HSIC Control Register (USBNC_USB_UH2_HSIC_CTRL)

The USB Host2 HSIC control register controls Host2 high speed IC configuration. These features are not directly related to the USB functionality, but control special features, interfacing on the USB ports, as well as power control.

Address: 218_4000h base + 810h offset = 218_4810h



USBNC_USB_UH2_HSIC_CTRL field descriptions

Field	Description
31 CLK_VLD	Indicating whether Host2 HSIC clock is valid.

Table continues on the next page...

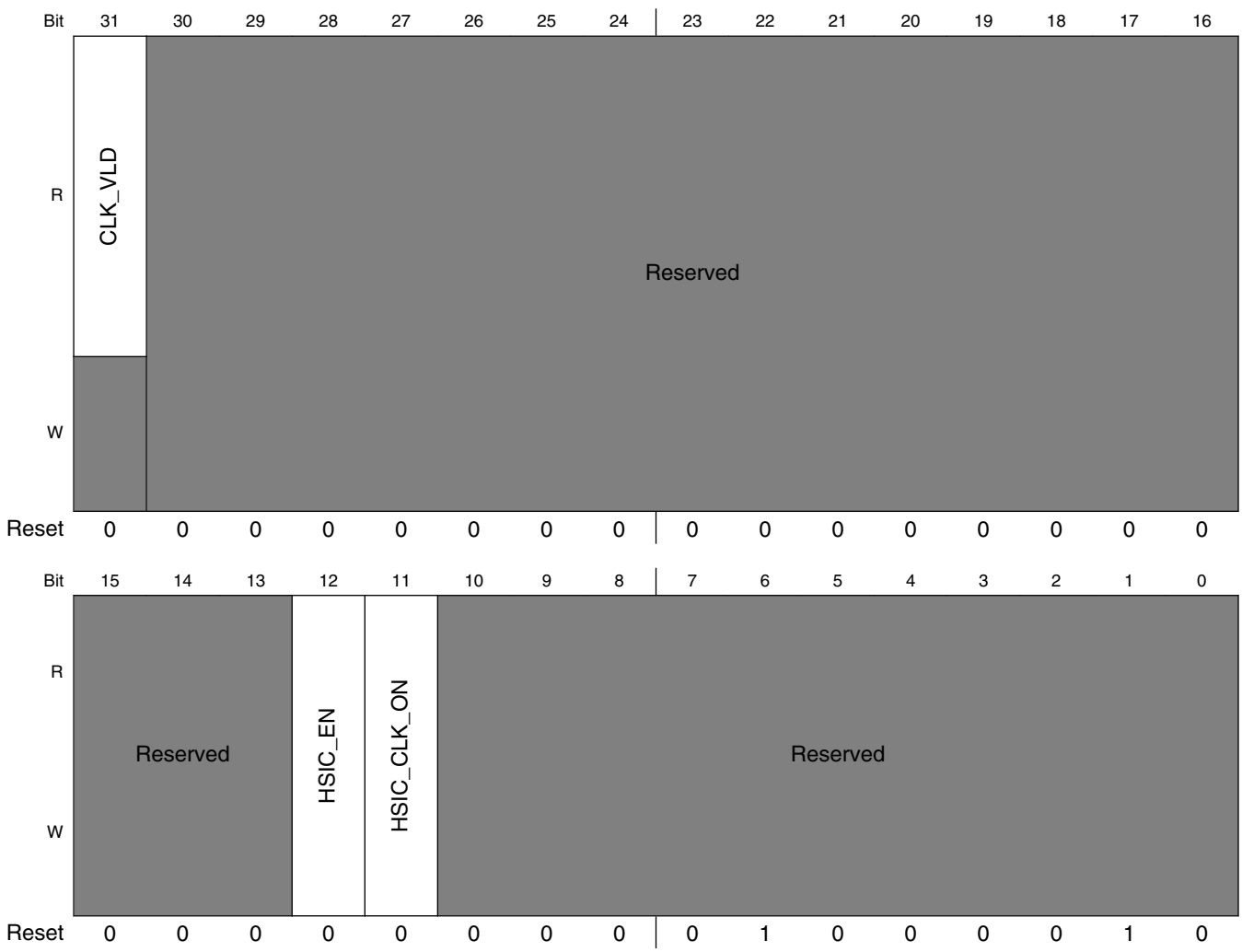
USBNC_USB_UH2_HSIC_CTRL field descriptions (continued)

Field	Description
	1 Valid 2 Invalid
30–13 -	This field is reserved. Reserved
12 HSIC_EN	Host2 HSIC enable 1 Enabled 0 Disabled
11 HSIC_CLK_ON	Force Host2 HSIC module 480M clock on, even when in Host 2 is in suspend mode. 1 Active 0 Inactive
-	This field is reserved. Reserved

65.5.6 USB Host3 HSIC Control Register (USBNC_USB_UH3_HSIC_CTRL)

The USB Host3 HSIC control register controls Host3 high speed IC configuration. These features are not directly related to the USB functionality, but control special features, interfacing on the USB ports, as well as power control.

Address: 218_4000h base + 814h offset = 218_4814h



USBNC_USB_UH3_HSIC_CTRL field descriptions

Field	Description
31 CLK_VLD	Indicating whether Host3 HSIC clock is valid.

Table continues on the next page...

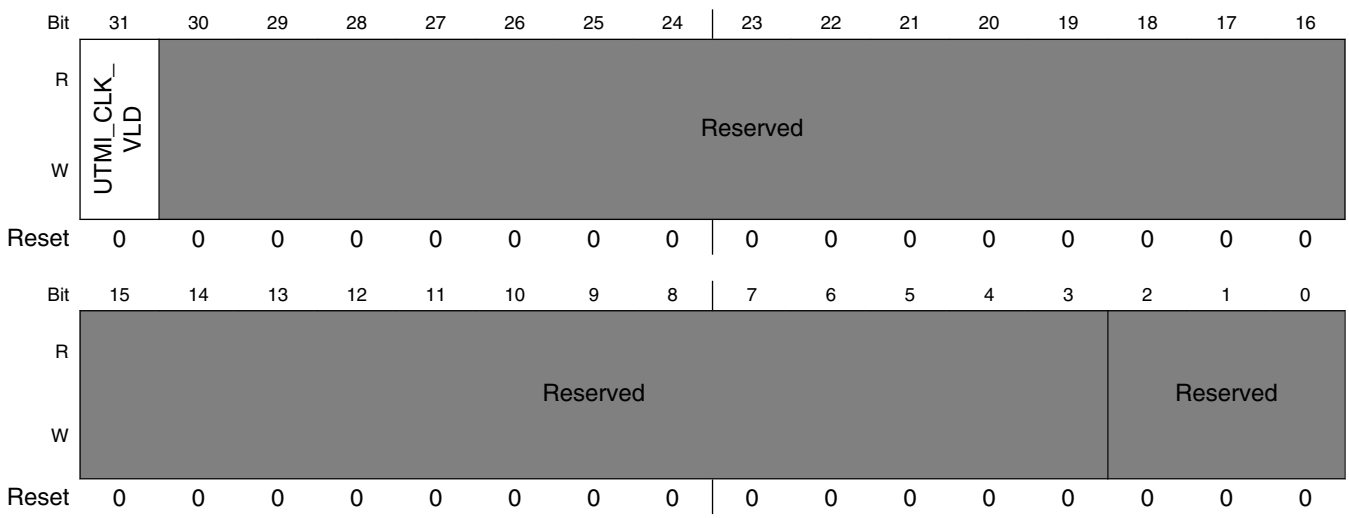
USBNC_USB_UH3_HSIC_CTRL field descriptions (continued)

Field	Description
	1 Valid 2 Invalid
30–13 -	This field is reserved. Reserved
12 HSIC_EN	Host3 HSIC enable 1 Enabled 0 Disabled
11 HSIC_CLK_ON	Force Host3 HSIC module 480M clock on, even when in Host 2 is in suspend mode. 1 Active 0 Inactive
-	This field is reserved. Reserved

65.5.7 OTG UTMI PHY Control 0 Register (USBNC_USB_OTG_PHY_CTRL_0)

USB OTG UTMI PHY control register 0 is used to control the on-chip OTG UTMI PHY.

Address: 218_4000h base + 818h offset = 218_4818h



USBNC_USB_OTG_PHY_CTRL_0 field descriptions

Field	Description
31 UTMI_CLK_VLD	Indicating whether OTG UTMI PHY clock is valid

Table continues on the next page...

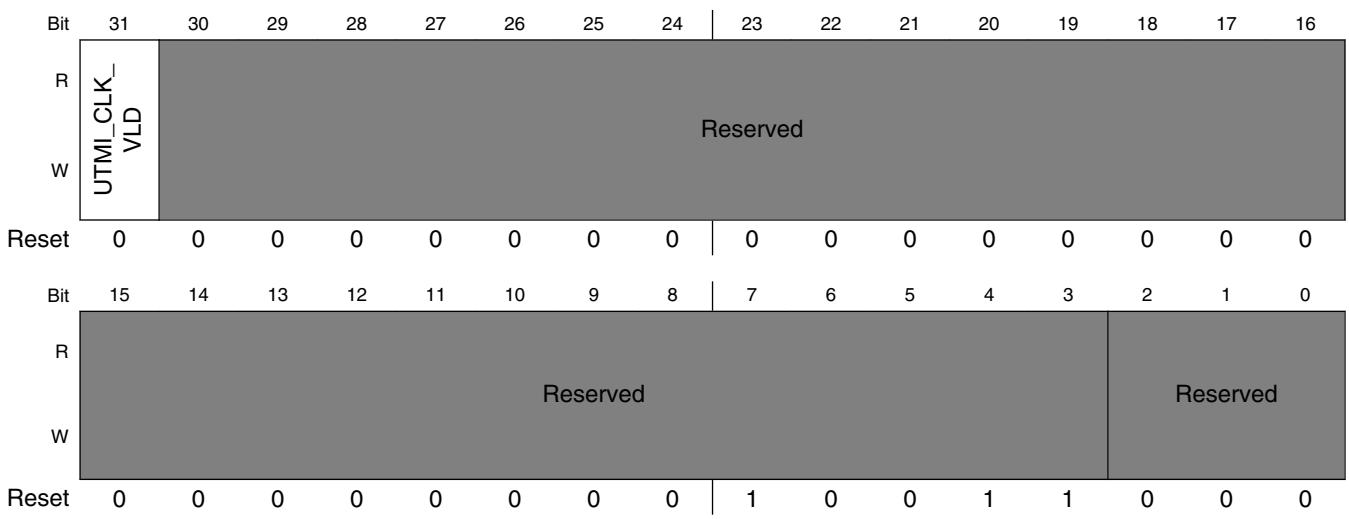
USBNC_USB_OTG_PHY_CTRL_0 field descriptions (continued)

Field	Description
	1 Valid 0 Invalid
30–3 -	This field is reserved. Reserved
-	This field is reserved. Reserved

65.5.8 Host1 UTMI PHY Control 0 Register (USBNC_USB_UH1_PHY_CTRL_0)

USB Host1 UTMI PHY Control Register 0 are used to control the on-chip Host1 UTMI PHY.

Address: 218_4000h base + 81Ch offset = 218_481Ch



USBNC_USB_UH1_PHY_CTRL_0 field descriptions

Field	Description
31 UTMI_CLK_VLD	Indicating whether Host 1 UTMI PHY clock is valid 1 Valid 0 Invalid
30–3 -	This field is reserved. Reserved
-	This field is reserved. Reserved

65.6 USB Core Memory Map/Register Definition

USBC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
218_4000	Identification register (USBC_UOG_ID)	32	R	E401_FA05h	65.6.1/5444
218_4004	Hardware General (USBC_UOG_HWGENERAL)	32	R	0000_0015h	65.6.2/5445
218_4008	Host Hardware Parameters (USBC_UOG_HWHOST)	32	R	1002_0001h	65.6.3/5446
218_400C	Device Hardware Parameters (USBC_UOG_HWDEVICE)	32	R	0000_0011h	65.6.4/5447
218_4010	TX Buffer Hardware Parameters (USBC_UOG_HWTXBUF)	32	R	8008_0B08h	65.6.5/5448
218_4014	RX Buffer Hardware Parameters (USBC_UOG_HWRXBUF)	32	R	0000_0808h	65.6.6/5448
218_4080	General Purpose Timer #0 Load (USBC_UOG_GPTIMER0LD)	32	R/W	0000_0000h	65.6.7/5449
218_4084	General Purpose Timer #0 Controller (USBC_UOG_GPTIMER0CTRL)	32	R/W	0000_0000h	65.6.8/5449
218_4088	General Purpose Timer #1 Load (USBC_UOG_GPTIMER1LD)	32	R/W	0000_0000h	65.6.9/5451
218_408C	General Purpose Timer #1 Controller (USBC_UOG_GPTIMER1CTRL)	32	R/W	0000_0000h	65.6.10/ 5451
218_4090	System Bus Config (USBC_UOG_SBUSCFG)	32	R/W	0000_0002h	65.6.11/ 5452
218_4100	Capability Registers Length (USBC_UOG_CAPLENGTH)	8	R	40h	65.6.12/ 5453
218_4102	Host Controller Interface Version (USBC_UOG_HCIVERSION)	16	R	0100h	65.6.13/ 5454
218_4104	Host Controller Structural Parameters (USBC_UOG_HCSPARAMS)	32	R	0001_0011h	65.6.14/ 5454
218_4108	Host Controller Capability Parameters (USBC_UOG_HCCPARAMS)	32	R	0000_0006h	65.6.15/ 5456
218_4120	Device Controller Interface Version (USBC_UOG_DCIVERSION)	16	R	0001h	65.6.16/ 5458
218_4124	Device Controller Capability Parameters (USBC_UOG_DCCPARAMS)	32	R	0000_0188h	65.6.17/ 5459
218_4140	USB Command Register (USBC_UOG_USBCMD)	32	R/W	0008_0000h	65.6.18/ 5460
218_4144	USB Status Register (USBC_UOG_USBSTS)	32	R/W	0000_0000h	65.6.19/ 5464
218_4148	Interrupt Enable Register (USBC_UOG_USBINTR)	32	R/W	0000_0000h	65.6.20/ 5468
218_414C	USB Frame Index (USBC_UOG_FRINDEX)	32	R/W	0000_0000h	65.6.21/ 5470
218_4154	Frame List Base Address (USBC_UOG_PERIODICLISTBASE)	32	R/W	0000_0000h	65.6.22/ 5471

Table continues on the next page...

USBC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
218_4154	Device Address (USBC_UOG_DEVICEADDR)	32	R/W	0000_0000h	65.6.23/ 5471
218_4158	Next Asynch. Address (USBC_UOG_ASYNCLISTADDR)	32	R/W	0000_0000h	65.6.24/ 5472
218_4158	Endpoint List Address (USBC_UOG_ENDPTLISTADDR)	32	R/W	0000_0000h	65.6.25/ 5473
218_4160	Programmable Burst Size (USBC_UOG_BURSTSIZE)	32	R/W	0000_0000h	65.6.26/ 5474
218_4164	TX FIFO Fill Tuning (USBC_UOG_TXFILLTUNING)	32	R/W	0000_0808h	65.6.27/ 5474
218_4178	Endpoint NAK (USBC_UOG_ENDPTNAK)	32	R/W	0000_0000h	65.6.28/ 5476
218_417C	Endpoint NAK Enable (USBC_UOG_ENDPTNAKEN)	32	R/W	0000_0000h	65.6.29/ 5476
218_4180	Configure Flag Register (USBC_UOG_CONFIGFLAG)	32	R/W	0000_0001h	65.6.30/ 5477
218_4184	Port Status & Control (USBC_UOG_PORTSC1)	32	R/W	1000_0000h	65.6.31/ 5478
218_41A4	On-The-Go Status & control (USBC_UOG_OTGSC)	32	R/W	0000_0120h	65.6.32/ 5484
218_41A8	USB Device Mode (USBC_UOG_USBMODE)	32	R/W	0000_0000h	65.6.33/ 5488
218_41AC	Endpoint Setup Status (USBC_UOG_ENDPTSETUPSTAT)	32	R/W	0000_0000h	65.6.34/ 5490
218_41B0	Endpoint Prime (USBC_UOG_ENDPTPRIME)	32	R/W	0000_0000h	65.6.35/ 5490
218_41B4	Endpoint Flush (USBC_UOG_ENDPTFLUSH)	32	R/W	0000_0000h	65.6.36/ 5491
218_41B8	Endpoint Status (USBC_UOG_ENDPTSTAT)	32	R	0000_0000h	65.6.37/ 5492
218_41BC	Endpoint Complete (USBC_UOG_ENDPTCOMPLETE)	32	R/W	0000_0000h	65.6.38/ 5493
218_41C0	Endpoint Control0 (USBC_UOG_ENDPTCTRL0)	32	R/W	0080_0080h	65.6.39/ 5494
218_41C4	Endpoint Control 1 (USBC_UOG_ENDPTCTRL1)	32	R/W	0000_0000h	65.6.40/ 5495
218_41C8	Endpoint Control 2 (USBC_UOG_ENDPTCTRL2)	32	R/W	0000_0000h	65.6.41/ 5498
218_41CC	Endpoint Control 3 (USBC_UOG_ENDPTCTRL3)	32	R/W	0000_0000h	65.6.42/ 5501
218_41D0	Endpoint Control 4 (USBC_UOG_ENDPTCTRL4)	32	R/W	0000_0000h	65.6.43/ 5503
218_41D4	Endpoint Control 5 (USBC_UOG_ENDPTCTRL5)	32	R/W	0000_0000h	65.6.44/ 5506

Table continues on the next page...

USBC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
218_41D8	Endpoint Control 6 (USBC_UOG_ENDPTCTRL6)	32	R/W	0000_0000h	65.6.45/ 5509
218_41DC	Endpoint Control 7 (USBC_UOG_ENDPTCTRL7)	32	R/W	0000_0000h	65.6.46/ 5511
218_4200	Identification register (USBC_UH1_ID)	32	R	E401_FA05h	65.6.1/5444
218_4204	Hardware General (USBC_UH1_HWGENERAL)	32	R	0000_0015h	65.6.2/5445
218_4208	Host Hardware Parameters (USBC_UH1_HWHOST)	32	R	1002_0001h	65.6.3/5446
218_4210	TX Buffer Hardware Parameters (USBC_UH1_HWTXBUF)	32	R	8008_0B08h	65.6.5/5448
218_4214	RX Buffer Hardware Parameters (USBC_UH1_HWRXBUF)	32	R	0000_0808h	65.6.6/5448
218_4280	General Purpose Timer #0 Load (USBC_UH1_GPTIMER0LD)	32	R/W	0000_0000h	65.6.7/5449
218_4284	General Purpose Timer #0 Controller (USBC_UH1_GPTIMER0CTRL)	32	R/W	0000_0000h	65.6.8/5449
218_4288	General Purpose Timer #1 Load (USBC_UH1_GPTIMER1LD)	32	R/W	0000_0000h	65.6.9/5451
218_428C	General Purpose Timer #1 Controller (USBC_UH1_GPTIMER1CTRL)	32	R/W	0000_0000h	65.6.10/ 5451
218_4290	System Bus Config (USBC_UH1_SBUSCFG)	32	R/W	0000_0002h	65.6.11/ 5452
218_4300	Capability Registers Length (USBC_UH1_CAPLENGTH)	8	R	40h	65.6.12/ 5453
218_4302	Host Controller Interface Version (USBC_UH1_HCIVERSION)	16	R	0100h	65.6.13/ 5454
218_4304	Host Controller Structural Parameters (USBC_UH1_HCSPARAMS)	32	R	0001_0011h	65.6.14/ 5454
218_4308	Host Controller Capability Parameters (USBC_UH1_HCCPARAMS)	32	R	0000_0006h	65.6.15/ 5456
218_4340	USB Command Register (USBC_UH1_USBCMD)	32	R/W	0008_0000h	65.6.18/ 5460
218_4344	USB Status Register (USBC_UH1_USBSTS)	32	R/W	0000_0000h	65.6.19/ 5464
218_4348	Interrupt Enable Register (USBC_UH1_USBINTR)	32	R/W	0000_0000h	65.6.20/ 5468
218_434C	USB Frame Index (USBC_UH1_FRINDEX)	32	R/W	0000_0000h	65.6.21/ 5470
218_4354	Frame List Base Address (USBC_UH1_PERIODICLISTBASE)	32	R/W	0000_0000h	65.6.22/ 5471
218_4358	Next Asynch. Address (USBC_UH1_ASYNCLISTADDR)	32	R/W	0000_0000h	65.6.24/ 5472
218_4360	Programmable Burst Size (USBC_UH1_BURSTSIZE)	32	R/W	0000_0000h	65.6.26/ 5474
218_4364	TX FIFO Fill Tuning (USBC_UH1_TXFILLTUNING)	32	R/W	0000_0808h	65.6.27/ 5474

Table continues on the next page...

USBC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
218_4380	Configure Flag Register (USBC_UH1_CONFIGFLAG)	32	R/W	0000_0001h	65.6.30/ 5477
218_4384	Port Status & Control (USBC_UH1_PORTSC1)	32	R/W	1000_0000h	65.6.31/ 5478
218_43A8	USB Device Mode (USBC_UH1_USBMODE)	32	R/W	0000_0000h	65.6.33/ 5488
218_4400	Identification register (USBC_UH2_ID)	32	R	E401_FA05h	65.6.1/5444
218_4404	Hardware General (USBC_UH2_HWGENERAL)	32	R	0000_0015h	65.6.2/5445
218_4408	Host Hardware Parameters (USBC_UH2_HWHOST)	32	R	1002_0001h	65.6.3/5446
218_4410	TX Buffer Hardware Parameters (USBC_UH2_HWTXBUF)	32	R	8008_0B08h	65.6.5/5448
218_4414	RX Buffer Hardware Parameters (USBC_UH2_HWRXBUF)	32	R	0000_0808h	65.6.6/5448
218_4480	General Purpose Timer #0 Load (USBC_UH2_GPTIMER0LD)	32	R/W	0000_0000h	65.6.7/5449
218_4484	General Purpose Timer #0 Controller (USBC_UH2_GPTIMER0CTRL)	32	R/W	0000_0000h	65.6.8/5449
218_4488	General Purpose Timer #1 Load (USBC_UH2_GPTIMER1LD)	32	R/W	0000_0000h	65.6.9/5451
218_448C	General Purpose Timer #1 Controller (USBC_UH2_GPTIMER1CTRL)	32	R/W	0000_0000h	65.6.10/ 5451
218_4490	System Bus Config (USBC_UH2_SBUSCFG)	32	R/W	0000_0002h	65.6.11/ 5452
218_4500	Capability Registers Length (USBC_UH2_CAPLENGTH)	8	R	40h	65.6.12/ 5453
218_4502	Host Controller Interface Version (USBC_UH2_HCIVERSION)	16	R	0100h	65.6.13/ 5454
218_4504	Host Controller Structural Parameters (USBC_UH2_HCSPARAMS)	32	R	0001_0011h	65.6.14/ 5454
218_4508	Host Controller Capability Parameters (USBC_UH2_HCCPARAMS)	32	R	0000_0006h	65.6.15/ 5456
218_4540	USB Command Register (USBC_UH2_USBCMD)	32	R/W	0008_0000h	65.6.18/ 5460
218_4544	USB Status Register (USBC_UH2_USBSTS)	32	R/W	0000_0000h	65.6.19/ 5464
218_4548	Interrupt Enable Register (USBC_UH2_USBINTR)	32	R/W	0000_0000h	65.6.20/ 5468
218_454C	USB Frame Index (USBC_UH2_FRINDEX)	32	R/W	0000_0000h	65.6.21/ 5470
218_4554	Frame List Base Address (USBC_UH2_PERIODICLISTBASE)	32	R/W	0000_0000h	65.6.22/ 5471
218_4558	Next Asynch. Address (USBC_UH2_ASYNCLISTADDR)	32	R/W	0000_0000h	65.6.24/ 5472
218_4560	Programmable Burst Size (USBC_UH2_BURSTSIZE)	32	R/W	0000_0000h	65.6.26/ 5474

Table continues on the next page...

USBC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
218_4564	TX FIFO Fill Tuning (USBC_UH2_TXFILLTUNING)	32	R/W	0000_0808h	65.6.27/ 5474
218_4580	Configure Flag Register (USBC_UH2_CONFIGFLAG)	32	R/W	0000_0001h	65.6.30/ 5477
218_4584	Port Status & Control (USBC_UH2_PORTSC1)	32	R/W	1000_0000h	65.6.31/ 5478
218_45A8	USB Device Mode (USBC_UH2_USBMODE)	32	R/W	0000_0000h	65.6.33/ 5488
218_4600	Identification register (USBC_UH3_ID)	32	R	E401_FA05h	65.6.1/5444
218_4604	Hardware General (USBC_UH3_HWGENERAL)	32	R	0000_0015h	65.6.2/5445
218_4608	Host Hardware Parameters (USBC_UH3_HWHOST)	32	R	1002_0001h	65.6.3/5446
218_4610	TX Buffer Hardware Parameters (USBC_UH3_HWTXBUF)	32	R	8008_0B08h	65.6.5/5448
218_4614	RX Buffer Hardware Parameters (USBC_UH3_HWRXBUF)	32	R	0000_0808h	65.6.6/5448
218_4680	General Purpose Timer #0 Load (USBC_UH3_GPTIMER0LD)	32	R/W	0000_0000h	65.6.7/5449
218_4684	General Purpose Timer #0 Controller (USBC_UH3_GPTIMER0CTRL)	32	R/W	0000_0000h	65.6.8/5449
218_4688	General Purpose Timer #1 Load (USBC_UH3_GPTIMER1LD)	32	R/W	0000_0000h	65.6.9/5451
218_468C	General Purpose Timer #1 Controller (USBC_UH3_GPTIMER1CTRL)	32	R/W	0000_0000h	65.6.10/ 5451
218_4690	System Bus Config (USBC_UH3_SBUSCFG)	32	R/W	0000_0002h	65.6.11/ 5452
218_4700	Capability Registers Length (USBC_UH3_CAPLENGTH)	8	R	40h	65.6.12/ 5453
218_4702	Host Controller Interface Version (USBC_UH3_HCIVERSION)	16	R	0100h	65.6.13/ 5454
218_4704	Host Controller Structural Parameters (USBC_UH3_HCSPARAMS)	32	R	0001_0011h	65.6.14/ 5454
218_4708	Host Controller Capability Parameters (USBC_UH3_HCCPARAMS)	32	R	0000_0006h	65.6.15/ 5456
218_4740	USB Command Register (USBC_UH3_USBCMD)	32	R/W	0008_0000h	65.6.18/ 5460
218_4744	USB Status Register (USBC_UH3_USBSTS)	32	R/W	0000_0000h	65.6.19/ 5464
218_4748	Interrupt Enable Register (USBC_UH3_USBINTR)	32	R/W	0000_0000h	65.6.20/ 5468
218_474C	USB Frame Index (USBC_UH3_FRINDEX)	32	R/W	0000_0000h	65.6.21/ 5470
218_4754	Frame List Base Address (USBC_UH3_PERIODICLISTBASE)	32	R/W	0000_0000h	65.6.22/ 5471
218_4758	Next Asynch. Address (USBC_UH3_ASYNCLISTADDR)	32	R/W	0000_0000h	65.6.24/ 5472

Table continues on the next page...

USBC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
218_4760	Programmable Burst Size (USBC_UH3_BURSTSIZE)	32	R/W	0000_0000h	65.6.26/ 5474
218_4764	TX FIFO Fill Tuning (USBC_UH3_TXFILLTUNING)	32	R/W	0000_0808h	65.6.27/ 5474
218_4780	Configure Flag Register (USBC_UH3_CONFIGFLAG)	32	R/W	0000_0001h	65.6.30/ 5477
218_4784	Port Status & Control (USBC_UH3_PORTSC1)	32	R/W	1000_0000h	65.6.31/ 5478
218_47A8	USB Device Mode (USBC_UH3_USBMODE)	32	R/W	0000_0000h	65.6.33/ 5488

65.6.1 Identification register (USBC_*n*_ID)

The ID register identifies the USB 2.0 High-Speed core and its revision.

Address: 218_4000h base + 0h offset + (512d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved								REVISION							
W																
Reset	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		NID						Reserved		ID					
W																
Reset	1	1	1	1	1	0	1	0	0	0	0	0	0	1	0	1

USBC_*n*_ID field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 REVISION	Revision number of the controller core.
15–14 -	This field is reserved. Reserved
13–8 NID	Complement version of ID
7–6 -	This field is reserved. Reserved
ID	Configuration number.

Table continues on the next page...

USBC_*n*_ID field descriptions (continued)

Field	Description
	This number is set to 0x05 and indicates that the peripheral is USB 2.0 High-Speed core.

65.6.2 Hardware General (USBC_*n*_HWGENERAL)

General hardware parameters as defined in System Level Issues and Core Configuration.

NOTE

The reset value could vary from instance to instance. Please see the detail in bit field description and ignore reset value in summary table in this case!

Address: 218_4000h base + 4h offset + (512d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved					SM		PHYM			PHYW		Reserved			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1

USBC_*n*_HWGENERAL field descriptions

Field	Description
31–11 -	This field is reserved. Reserved
10–9 SM	Serial interface mode capability SM bit reset value is '00b' 00 No Serial Engine, always use parallel signalling. 01 Serial Engine present, always use serial signalling for FS/LS. 10 Software programmable - Reset to use parallel signalling for FS/LS 11 Software programmable - Reset to use serial signalling for FS/LS
8–6 PHYM	Transciever type PHYM bit reset value: '0000b' for OTG controller core, '0100b' for Host-only controller core. 000 UTMI/UMTI+ 001 ULPI DDR 010 ULPI 011 Serial Only 100 Software programmable - reset to UTMI/UTMI+ 101 Software programmable - reset to ULPI DDR

Table continues on the next page...

USBC_*n*_HWGENERAL field descriptions (continued)

Field	Description
	110 Software programmable - reset to ULPI 111 Software programmable - reset to Serial 1000 IC-USB 1001 Software programmable - reset to IC-USB 1010 HSIC 1011 Software programmable - reset to HSIC
5–4 PHYW	Data width of the transciever connected to the controller core. PHYW bit reset value is '01b'. 00 8 bit wide data bus Software non-programmable 01 16 bit wide data bus Software non-programmable 10 Reset to 8 bit wide data bus Software programmable 11 Reset to 16 bit wide data bus Software programmable
-	This field is reserved. Reserved

65.6.3 Host Hardware Parameters (USBC_*n*_HWHOST)

Address: 218_4000h base + 8h offset + (512d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved												NPORT			HC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

USBC_*n*_HWHOST field descriptions

Field	Description
31–4 -	This field is reserved. Reserved
3–1 NPORT	The Nmber of downstream ports supported by the host controller is (NPORT+1). **NOTE:** When these bits value is '000', it indicates a single-port host controller.

Table continues on the next page...

USBC_*n*_HWHOST field descriptions (continued)

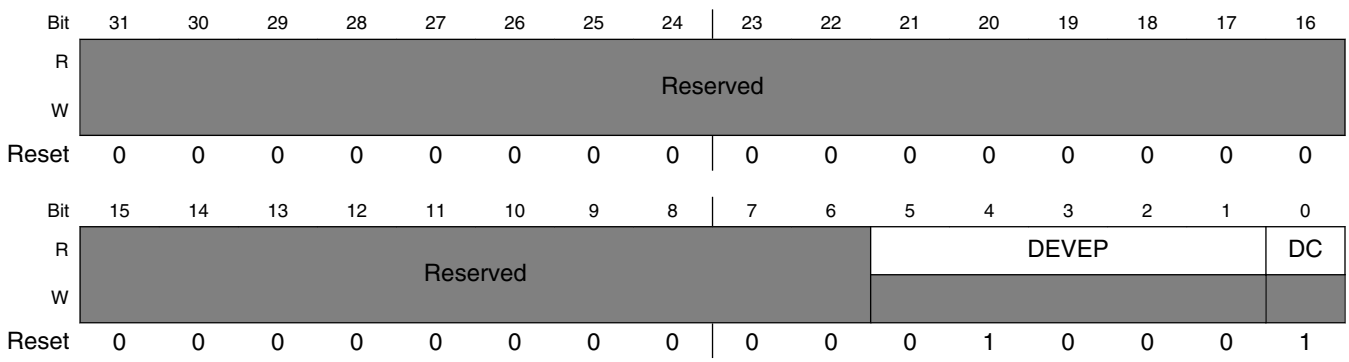
Field	Description
0 HC	Host Capable. Indicating whether host operation mode is supported or not. 1 Supported 0 Not supported

65.6.4 Device Hardware Parameters (USBC_*n*_HWDEVICE)

NOTE

This register is only available in OTG core.

Address: 218_4000h base + Ch offset + (512d × i), where i=0d to 0d



USBC_*n*_HWDEVICE field descriptions

Field	Description
31–6 -	This field is reserved. Reserved
5–1 DEVEP	Device Endpoint Number
0 DC	Device Capable. Indicating whether device operation mode is supported or not. 1 Supported 0 Not supported

65.6.5 TX Buffer Hardware Parameters (USBC_*n*_HWTXBUF)

Address: 218_4000h base + 10h offset + (512d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								TXCHANADD								Reserved								TXBURST							
W																																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	1	0	0	0

USBC_*n*_HWTXBUF field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 TXCHANADD	TX FIFO Buffer size is: (2^TXCHANADD) * 4 Bytes. These bits are set to '08h', so buffer size is 256*4 Bytes. For the OTG controller operating in device mode, this is the FIFO buffer size per endpoint. As the OTG controller has 8 TX endpoint, there are 8 of these buffers. For the OTG controller operating in host mode, or for Host-only controller, the entire buffer memory is used as a single TX buffer. Therefore, there is only 1 of this buffer
15–8 -	This field is reserved. Reserved
TXBURST	Default burst size for memory to TX buffer transfer. This is reset value of TXPBURST bits in USB core regsiter USB_n_BURSTSIZE. Please see Programmable Burst Size (USBC_*n*_BURSTSIZE) .

65.6.6 RX Buffer Hardware Parameters (USBC_*n*_HWRXBUF)

Address: 218_4000h base + 14h offset + (512d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved																RXADD								RXBURST							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0

USBC_*n*_HWRXBUF field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
15–8 RXADD	Buffer total size for all receive endpoints is (2^RXADD). RX Buffer size is: (2^RXADD) * 4 Bytes. These bits are set to '08h', so buffer size is 256*4 Bytes.

Table continues on the next page...

USBC_*n*_HWRXBUF field descriptions (continued)

Field	Description
	There is a single Receive FIFO buffer in the USB controller. The buffer is shared for all endpoints for the OTG controller in device mode.
RXBURST	Default burst size for memory to RX buffer transfer. This is reset value of RXPBURST bits in USB core regsiter USB_n_BURSTSIZE. Please see Programmable Burst Size (USBC_*n*_BURSTSIZE) .

65.6.7 General Purpose Timer #0 Load (USBC_*n*_GPTIMER0LD)

This register controls load value of the count timer in register n_GPTIMER0CTRL. Please see General Purpose Timer #0 Controller (USBC_*n*_GPTIMER0CTRL) .

Address: 218_4000h base + 80h offset + (512d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								GPTLD																							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USBC_*n*_GPTIMER0LD field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
GPTLD	General Purpose Timer Load Value These bit fields are loaded to GPTCNT bits when GPTRST bit is set '1b'. This value represents the time in microseconds minus 1 for the timer duration. Example: for a one millisecond timer, load 1000-1=999 or 0x0003E7. **NOTE:** Max value is 0xFFFFFF or 16.777215 seconds.

65.6.8 General Purpose Timer #0 Controller (USBC_*n*_GPTIMER0CTRL)

This register contains the control for this countdown timer and a data field can be queried to determine the running count value. This timer has granularity on 1 us and can be programmed to a little over 16 seconds. There are two counter modes which are described in the register table below. When the timer counter value transitions to zero, an interrupt could be generated if enable.

Interrupt status bit is TI0 bit in n_USBSTS register (See USB Status Register (USBC_*n*_USBSTS)), interrupt enable bit is TIE0 bit in n_USBINTR register. (See Interrupt Enable Register (USBC_*n*_USBINTR) .)

Address: 218_4000h base + 84h offset + (512d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	GPTRUN	GPTRST	Reserved					GPTMODE	GPTCNT							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GPTCNT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USBC_*n*_GPTIMER0CTRL field descriptions

Field	Description
31 GPTRUN	General Purpose Timer Run GPTCNT bits are not effected when setting or clearing this bit. 0 Stop counting 1 Run
30 GPTRST	General Purpose Timer Reset 0 No action 1 Load counter value from GPTLD bits in n_GPTIMER0LD
29–25 -	This field is reserved. Reserved
24 GPTMODE	General Purpose Timer Mode In one shot mode, the timer will count down to zero, generate an interrupt, and stop until the counter is reset by software; In repeat mode, the timer will count down to zero, generate an interrupt and automatically reload the counter value from GPTLD bits to start again. 0 One Shot Mode 1 Repeat Mode
GPTCNT	General Purpose Timer Counter. This field is the count value of the countdown timer.

65.6.9 General Purpose Timer #1 Load (USBC_*n*_GPTIMER1LD)

This register controls load value of the count timer in register n_GPTIMER1CTRL. Please see General Purpose Timer #1 Controller (USBC_*n*_GPTIMER1CTRL) .

Address: 218_4000h base + 88h offset + (512d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								GPTLD																							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USBC_*n*_GPTIMER1LD field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
GPTLD	General Purpose Timer Load Value These bit fields are loaded to GPTCNT bits when GPTRST bit is set '1b'. This value represents the time in microseconds minus 1 for the timer duration. Example: for a one millisecond timer, load 1000-1=999 or 0x0003E7. **NOTE:** Max value is 0xFFFFFF or 16.777215 seconds.

65.6.10 General Purpose Timer #1 Controller (USBC_*n*_GPTIMER1CTRL)

This register contains the control for this countdown timer and a data field can be queried to determine the running count value. This timer has granularity on 1 us and can be programmed to a little over 16 seconds. There are two counter modes which are described in the register table below. When the timer counter value transitions to zero, an interrupt could be generated if enable.

Interrupt status bit is TI1 bit in USB_n_USBSTS register (See USB Status Register (USBC_*n*_USBSTS)), interrupt enable bit is TIE1 bit in n_USBINTR register (See Interrupt Enable Register (USBC_*n*_USBINTR)).

Address: 218_4000h base + 8Ch offset + (512d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	GPTRUN	GPTRST	Reserved					GPTMODE	GPTCNT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	GPTCNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USBC_*n*_GPTIMER1CTRL field descriptions

Field	Description
31 GPTRUN	General Purpose Timer Run GPTCNT bits are not effected when setting or clearing this bit. 0 Stop counting 1 Run
30 GPTRST	General Purpose Timer Reset 0 No action 1 Load counter value from GPTLD bits in USB_n_GPTIMER0LD
29–25 -	This field is reserved. Reserved
24 GPTMODE	General Purpose Timer Mode In one shot mode, the timer will count down to zero, generate an interrupt, and stop until the counter is reset by software. In repeat mode, the timer will count down to zero, generate an interrupt and automatically reload the counter value from GPTLD bits to start again. 0 One Shot Mode 1 Repeat Mode
GPTCNT	General Purpose Timer Counter. This field is the count value of the countdown timer.

65.6.11 System Bus Config (USBC_*n*_SBUSCFG)

Address: 218_4000h base + 90h offset + (512d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved																													AHBBRST		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

USBC_*n*_SBUSCFG field descriptions

Field	Description
31–3 -	This field is reserved. Reserved
AHBBRST	AHB master interface Burst configuration These bits control AHB master transfer type sequence (or priority). **NOTE:** This register overrides n_BURSTSIZE register when its value is not zero. 000 Incremental burst of unspecified length only 001 INCR4 burst, then single transfer 010 INCR8 burst, INCR4 burst, then single transfer 011 INCR16 burst, INCR8 burst, INCR4 burst, then single transfer 100 Reserved, don't use 101 INCR4 burst, then incremental burst of unspecified length 110 INCR8 burst, INCR4 burst, then incremental burst of unspecified length 111 INCR16 burst, INCR8 burst, INCR4 burst, then incremental burst of unspecified length

65.6.12 Capability Registers Length (USBC_*n*_CAPLENGTH)

The Capability Registers Length register contains the address offset to the Operational registers relative to the CAPLENGTH register.

Address: 218_4000h base + 100h offset + (512d × i), where i=0d to 3d

Bit	7	6	5	4	3	2	1	0
Read	CAPLENGTH							
Write								
Reset	0	1	0	0	0	0	0	0

USBC_*n*_CAPLENGTH field descriptions

Field	Description
CAPLENGTH	These bits are used as an offset to add to register base to find the beginning of the Operational Register. Default value is '40h'.

65.6.13 Host Controller Interface Version (USBC_*n*_HCIVERSION)

This is a 2-byte register containing 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.

Address: 218_4000h base + 102h offset + (512d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	HCIVERSION															
Write																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

USBC_*n*_HCIVERSION field descriptions

Field	Description
HCIVERSION	Host Controller Interface Version Number Default value is '10h', which means EHCI rev1.0.

65.6.14 Host Controller Structural Parameters (USBC_*n*_HCSPARAMS)

The following figure shows the port steering logic capabilities of Host Control Structural Parameters (n_HCSPARAMS).

Address: 218_4000h base + 104h offset + (512d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				N_TT				N_PTT				Reserved			PI
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	N_CC				N_PCC				Reserved			PPC	N_PORTS			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

USBC_*n*_HCSPARAMS field descriptions

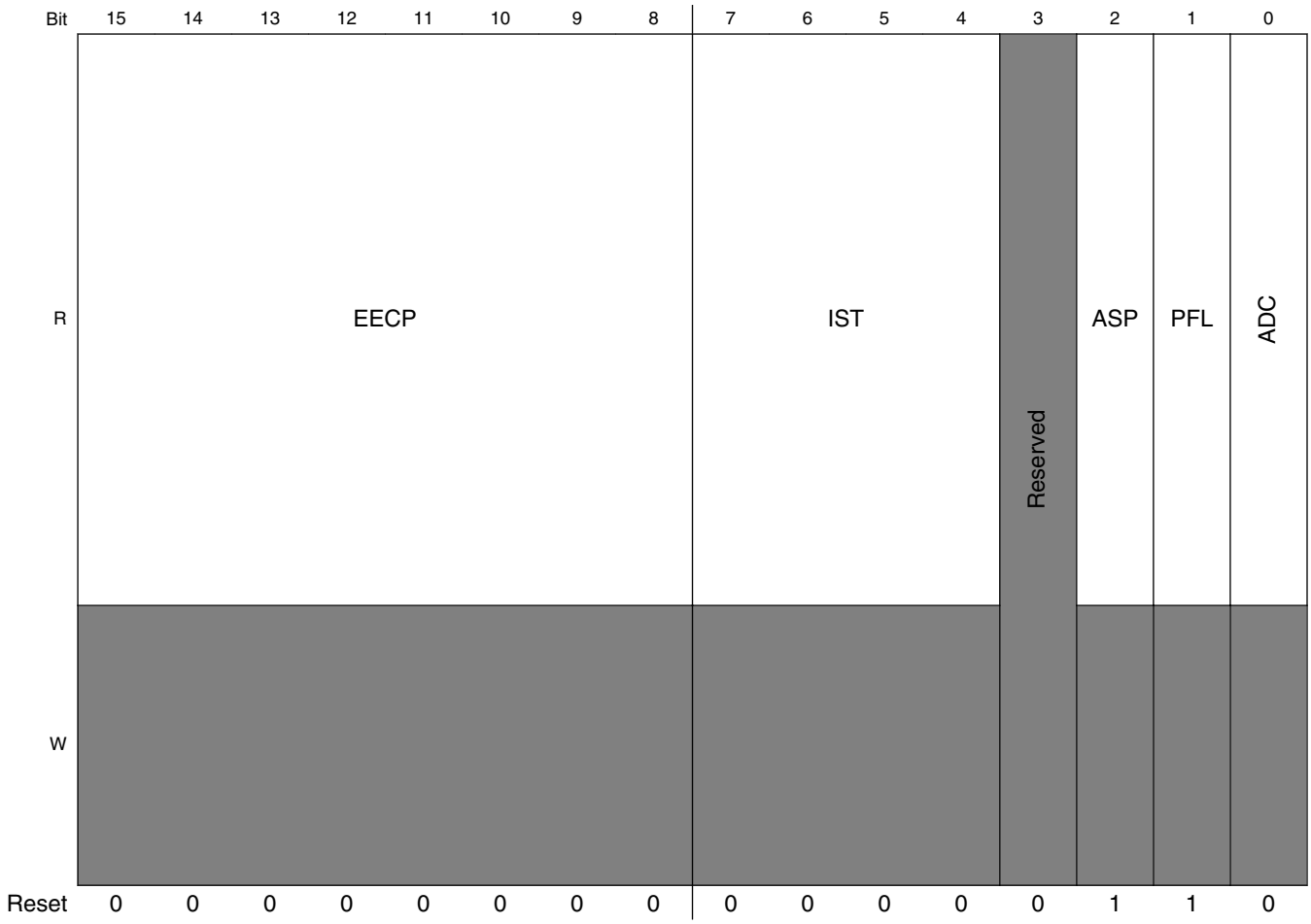
Field	Description
31–28 -	This field is reserved. Reserved
27–24 N_TT	Number of Transaction Translators (N_TT). Default value '0000b' This field indicates the number of embedded transaction translators associated with the USB2.0 host controller. These bits would be set to '0001b' for Multi-Port Host, and '0000b' for Single-Port Host.
23–20 N_PTT	Number of Ports per Transaction Translator (N_PTT). Default value '0000b' This field indicates the number of ports assigned to each transaction translator within the USB2.0 host controller. These bits would be set to equal N_PORTS for Multi-Port Host, and '0000b' for Single-Port Host.
19–17 -	This field is reserved. Reserved
16 PI	Port Indicators (P INDICATOR) This bit indicates whether the ports support port indicator control. When set to one, the port status and control registers include a read/writeable field for controlling the state of the port indicator This bit is "1b" in all controller core.
15–12 N_CC	Number of Companion Controller (N_CC). This field indicates the number of companion controllers associated with this USB2.0 host controller. These bits are '0000b' in all controller core. 0 There is no internal Companion Controller and port-ownership hand-off is not supported. 1 There are internal companion controller(s) and port-ownership hand-offs is supported.
11–8 N_PCC	Number of Ports per Companion Controller This field indicates the number of ports supported per internal Companion Controller. It is used to indicate the port routing configuration to the system software. For example, if N_PORTS has a value of 6 and N_CC has a value of 2 then N_PCC could have a value of 3. The convention is that the first N_PCC ports are assumed to be routed to companion controller 1, the next N_PCC ports to companion controller 2, etc. In the previous example, the N_PCC could have been 4, where the first 4 are routed to companion controller 1 and the last two are routed to companion controller 2. The number in this field must be consistent with N_PORTS and N_CC. These bits are '0000b' in all controller core.
7–5 -	This field is reserved. Reserved
4 PPC	Port Power Control This field indicates whether the host controller implementation includes port power control. A one indicates the ports have port power switches. A zero indicates the ports do not have port power switches. The value of this field affects the functionality of the Port Power field in each port status and control register
N_PORTS	Number of downstream ports. This field specifies the number of physical downstream ports implemented on this host controller. The value of this field determines how many port registers are addressable in the Operational Register. Valid values are in the range of 1h to Fh. A zero in this field is undefined. These bits are always set to '0001b' because all controller cores are Single-Port Host.

65.6.15 Host Controller Capability Parameters (USBC_*n*_HCCPARAMS)

This register identifies multiple mode control (time-base bit functionality), addressing capability.

Address: 218_4000h base + 108h offset + (512d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



USBC_*n*_HCCPARAMS field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
15–8 EECP	EHCI Extended Capabilities Pointer. This field indicates the existence of a capabilities list. A value of 00h indicates no extended capabilities are implemented. A non-zero value in this register indicates the offset in PCI configuration space of the first EHCI extended capability. The pointer value must be 40h or greater if implemented to maintain the consistency of the PCI header defined for this class of device. **NOTE:** These bits are set '00h' in all controller core.
7–4 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. These bits are set '00h' in all controller core.
3 -	This field is reserved. Reserved

Table continues on the next page...

USBC_*n*_HCCPARAMS field descriptions (continued)

Field	Description
2 ASP	Asynchronous Schedule Park Capability If this bit is set to a one, then 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. **NOTE:** ASP bit reset value: '00b' for OTG controller core, '11b' for Host-only controller core.
1 PFL	Programmable Frame List Flag If this bit is set to zero, then the 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. If set to a one, then the 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 bit is set '1b' in all controller core.
0 ADC	64-bit Addressing Capability This bit is set '0b' in all controller core, no 64-bit addressing capability is supported.

65.6.16 Device Controller Interface Version (USBC_*n*_DCIVERSION)

This register indicates the two-byte BCD encoding of the device controller interface version number.

Address: 218_4000h base + 120h offset + (512d × i), where i=0d to 0d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	DCIVERSION															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

USBC_*n*_DCIVERSION field descriptions

Field	Description
DCIVERSION	Device Controller Interface Version Number Default value is '01h', which means rev0.1.

65.6.17 Device Controller Capability Parameters (USBC_*n*_DCCPARAMS)

These fields describe the overall device capability of the controller.

NOTE

This register is only available in OTG controller core.

Address: 218_4000h base + 124h offset + (512d × i), where i=0d to 0d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved							HC	DC	Reserved		DEN				
W																
Reset	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0

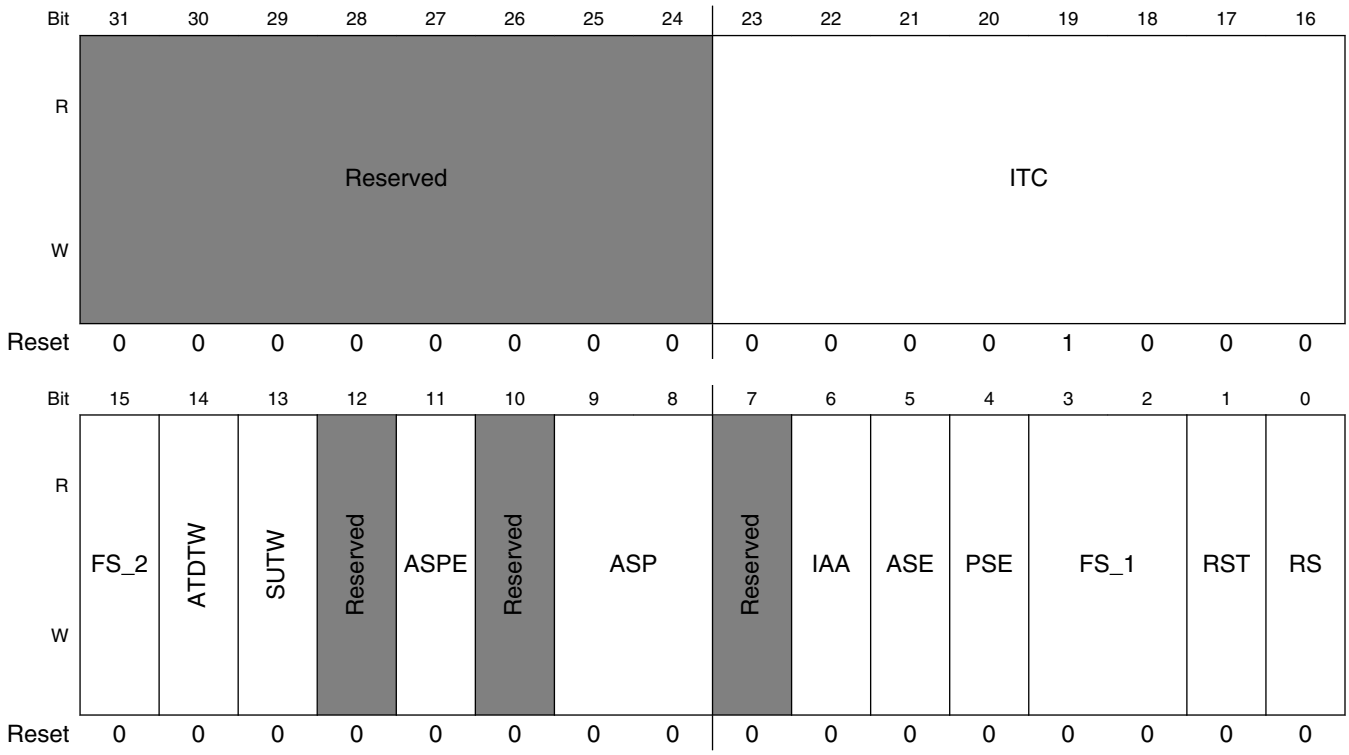
USBC_*n*_DCCPARAMS field descriptions

Field	Description
31–9 -	This field is reserved. Reserved
8 HC	Host Capable When this bit is 1, this controller is capable of operating as an EHCI compatible USB 2.0 host controller.
7 DC	Device Capable When this bit is 1, this controller is capable of operating as a USB 2.0 device.
6–5 -	This field is reserved. Reserved
DEN	Device Endpoint Number This field indicates the number of endpoints built into the device controller. If this controller is not device capable, then this field will be zero. Valid values are 0 - 15.

65.6.18 USB Command Register (USBC_*n*_USBCMD)

The Command Register indicates the command to be executed by the serial bus host/ device controller. Writing to the register causes a command to be executed.

Address: 218_4000h base + 140h offset + (512d × i), where i=0d to 3d



USBC_*n*_USBCMD field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 ITC	Interrupt Threshold Control -Read/Write. 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 0x00 Immediate (no threshold) 0x01 1 micro-frame 0x02 2 micro-frames 0x04 4 micro-frames 0x08 8 micro-frames 0x10 16 micro-frames

Table continues on the next page...

USBC_*n*_USBCMD field descriptions (continued)

Field	Description
	0x20 32 micro-frames 0x40 64 micro-frames
15 FS_2	See also bits 3-2 Frame List Size - (Read/Write or Read Only). [host mode only] This field is Read/Write only if Programmable Frame List Flag in the HCCPARAMS registers is set to one. 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:** This field is made up from USBCMD bits 15, 3 and 2. Value Meaning 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)
14 ATDTW	Add dTD TripWire - Read/Write. [device mode only] This bit is used as a semaphore to ensure proper addition of a new dTD to an active (primed) endpoint's linked list. This bit is set and cleared by software. This bit would also be cleared by hardware when state machine is hazard region for which adding a dTD to a primed endpoint may go unrecognized.
13 SUTW	Setup TripWire - Read/Write. [device mode only] This bit is used as a semaphore to ensure that the setup data payload of 8 bytes is extracted from a QH by the DCD without being corrupted. If the setup lockout mode is off (SLOM bit in USB core register n_USBMODE, see USB Device Mode (USBC_*n*_USBMODE)) then there is a hazard when new setup data arrives while the DCD is copying the setup data payload from the QH for a previous setup packet. This bit is set and cleared by software. This bit would also be cleared by hardware when a hazard detected.
12 -	This field is reserved. Reserved
11 ASPE	Asynchronous Schedule Park Mode Enable - Read/Write. If the *Asynchronous Park Capability* bit in the HCCPARAMS register is a one, then this bit defaults to a 1h and is R/W. Otherwise the bit must be a zero and is RO. 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. **NOTE:** ASPE bit reset value: '0b' for OTG controller core, '1b' for Host-only controller core.
10 -	This field is reserved. Reserved
9–8 ASP	Asynchronous Schedule Park Mode Count - 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 Read-Only. 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

Table continues on the next page...

USBC_*n*_USBCMD field descriptions (continued)

Field	Description
	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 all controller core.
7 -	This field is reserved. Reserved
6 IAA	Interrupt on Async Advance Doorbell - Read/Write. This bit is used as a doorbell by software to tell the host controller to issue an interrupt the next time it advances asynchronous schedule. Software must write a 1 to this bit to ring the 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.
5 ASE	Asynchronous Schedule Enable - Read/Write. Default 0b. This bit controls whether the host controller skips processing the Asynchronous Schedule. Only the host controller uses this bit. Values Meaning 0 Do not process the Asynchronous Schedule. 1 Use the ASYNCLISTADDR register to access the Asynchronous Schedule.
4 PSE	Periodic Schedule Enable- Read/Write. Default 0b. This bit controls whether the host controller skips processing the Periodic Schedule. Only the host controller uses this bit. Values Meaning 0 Do not process the Periodic Schedule 1 Use the PERIODICLISTBASE register to access the Periodic Schedule.
3–2 FS_1	See description at bit 15
1 RST	Controller Reset (RESET) - Read/Write. 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 operation mode: When software writes a one to this bit, the 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 will result in undefined behavior. Device operation mode:

Table continues on the next page...

USBC_*n*_USBCMD field descriptions (continued)

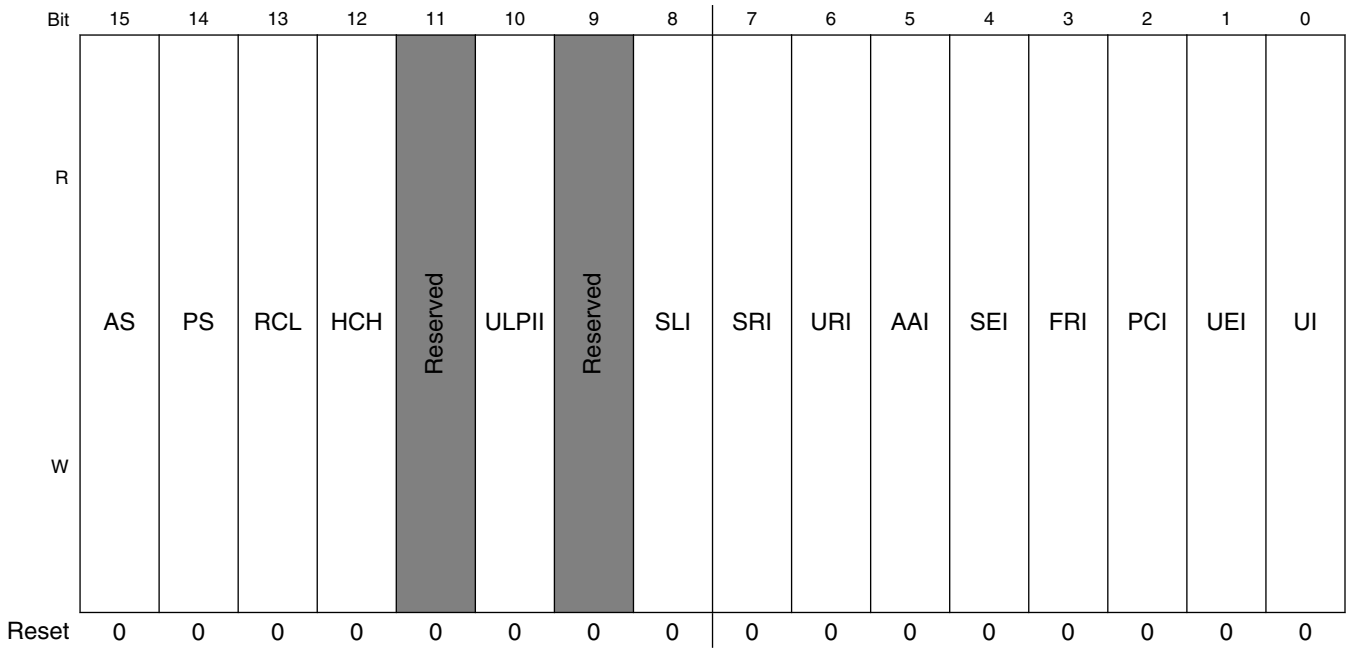
Field	Description
	When software writes a one to this bit, the Controller resets its internal pipelines, timers, counters, state machines etc. to their initial value. Writing a one to this bit when the device is in the attached state is not recommended, because the effect on an attached host is undefined. In order to ensure that the device is not in an attached state before initiating a device controller reset, all primed endpoints should be flushed and the USBCMD Run/Stop bit should be set to 0.
0 RS	Run/Stop (RS) - Read/Write. Default 0b. 1=Run. 0=Stop. Host operation mode: When set to '1b', the Controller proceeds with the execution of the schedule. The 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 HC Halted bit in the status register indicates when the Controller has finished the transaction and has entered the stopped state. Software should not write a one to this field unless the controller is in the Halted state (that is, HCHalted in the USBSTS register is a one). Device operation mode: Writing a one to this bit will cause the 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 controller has been properly initialized. Writing a 0 to this will cause a detach event.

65.6.19 USB Status Register (USBC_*n*_USBSTS)

This register indicates various states of the Host/Device Controller and any pending interrupts. This register does not indicate status resulting from a transaction on the serial bus.

Address: 218_4000h base + 144h offset + (512d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved						TI1	TI0	Reserved							NAKI
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



USBC_*n*_USBSTS field descriptions

Field	Description
31–26 -	This field is reserved. Reserved
25 TI1	General Purpose Timer Interrupt 1(GPTINT1)--R/WC. This bit is set when the counter in the GPTIMER1CTRL register transitions to zero, writing a one to this bit will clear it.
24 TI0	General Purpose Timer Interrupt 0(GPTINT0)--R/WC. This bit is set when the counter in the GPTIMER0CTRL register transitions to zero, writing a one to this bit clears it.
23–17 -	This field is reserved. Reserved
16 NAKI	NAK Interrupt Bit--RO. This bit is set by hardware when for a particular endpoint both the TX/RX Endpoint NAK bit and corresponding TX/RX Endpoint NAK Enable bit are set. This bit is automatically cleared by hardware when all Enabled TX/RX Endpoint NAK bits are cleared.
15 AS	Asynchronous Schedule Status - Read Only. 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. When this bit and the Asynchronous Schedule Enable bit are the same value, the Asynchronous Schedule is either enabled (1) or disabled (0). Only used in the host operation mode.
14 PS	Periodic Schedule Status - Read Only.

Table continues on the next page...

USBC_*n*_USBSTS field descriptions (continued)

Field	Description
	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. When this bit and the Periodic Schedule Enable bit are the same value, the Periodic Schedule is either enabled (1) or disabled (0). Only used in the host operation mode.
13 RCL	Reclamation - Read Only. This is a read-only status bit used to detect an empty asynchronous schedule. Only used in the host operation mode.
12 HCH	HCHalted - Read Only. This bit is a zero whenever the Run/Stop bit is a one. The 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 Controller hardware (for example, an internal error). Only used in the host operation mode. Default value is '0b' for OTG core, and '1b' for Host1/Host2/Host3 core. This is because OTG core is not operating as host in default. Please see CM bit in USB_n_USBMODE register. **NOTE:** HCH bit reset value: '0b' for OTG controller core, '1b' for Host-only controller core.
11 -	This field is reserved. Reserved
10 ULPII	ULPI Interrupt - R/WC. This bit will be set '1b' by hardware when there is an event completion in ULPI viewport. This bit is usable only if the controller support UPLI interface mode.
9 -	This field is reserved. Reserved
8 SLI	DCSuspend - R/WC. When a controller enters a suspend state from an active state, this bit will be set to a one. The device controller clears the bit upon exiting from a suspend state. Only used in device operation mode.
7 SRI	SOF Received - R/WC. 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 125ms in HS mode and will be synchronized to the actual SOF that is received. Because the device controller is initialized to FS before connect, this bit will be set at an interval of 1ms during the prelude to 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.
6 URI	USB Reset Received - R/WC. When the device controller detects a USB Reset and enters the default state, this bit will be set to a one. Software can write a 1 to this bit to clear the USB Reset Received status bit. Only used in device operation mode.

Table continues on the next page...

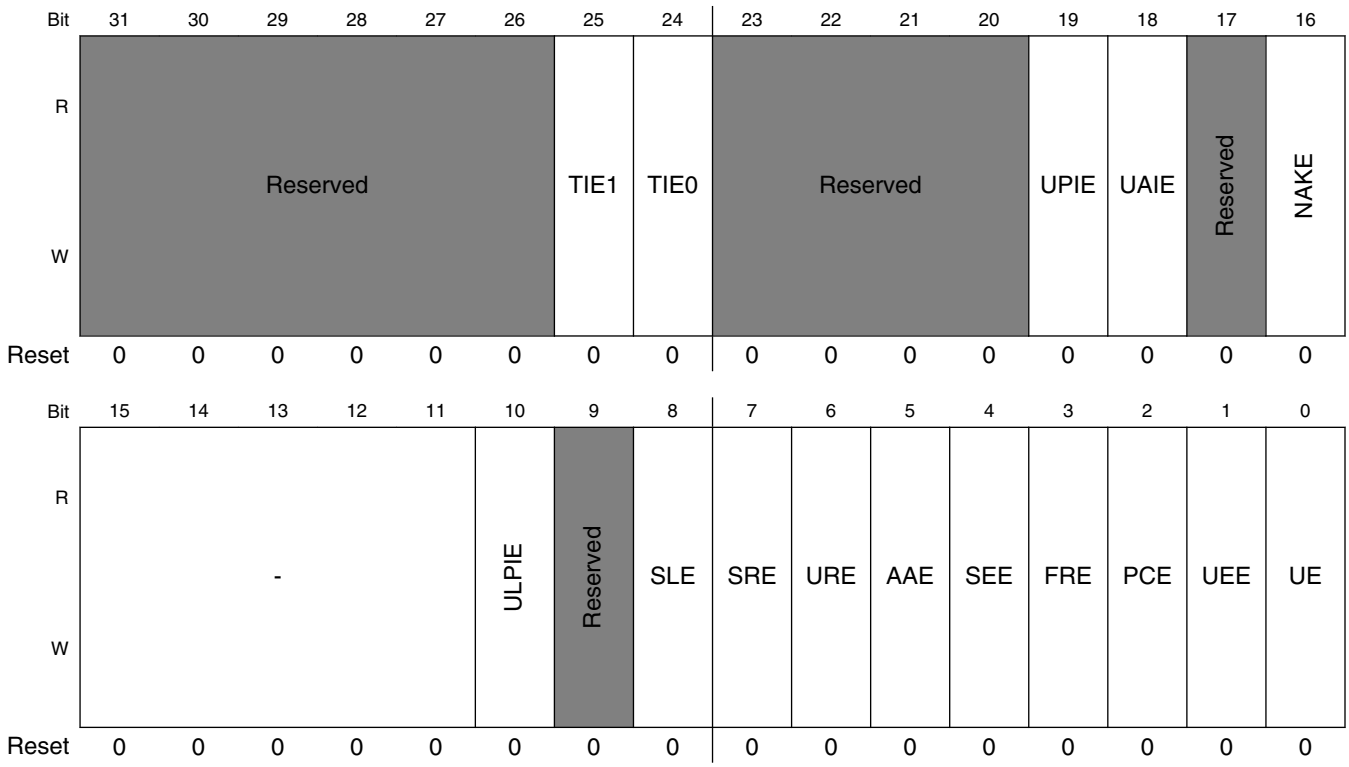
USBC_*n*_USBSTS field descriptions (continued)

Field	Description
5 AAI	Interrupt on Async Advance - R/WC. 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 n_USBCMD register. This status bit indicates the assertion of that interrupt source. Only used in host operation mode.
4 SEI	System Error- R/WC. This bit is will be set to '1b' when an Error response is seen to a read on the system interface.
3 FRI	Frame List Rollover - R/WC. The Host Controller sets this bit to a one when the Frame List Index rolls over from its maximum value to zero. 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 USB_n_USBCMD register) is 1024, the Frame Index Register rolls over every time FRINDEX [13] toggles. Similarly, if the size is 512, the Host Controller sets this bit to a one every time FHINDEX [12] toggles. Only used in host operation mode.
2 PCI	Port Change Detect - R/WC. The Host Controller sets this bit to a one 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 operation states due to Reset or Suspend events, the notification mechanisms are the USB Reset Received bit and the DCSuspend bits respectively.
1 UEI	USB Error Interrupt (USBERRINT) - R/WC. When completion of a USB transaction results in an error condition, this bit is set by the Host/Device Controller. This bit is set along with the USBINT bit, if the TD on which the error interrupt occurred also had its interrupt on complete (IOC) bit set The device controller detects resume signaling only.
0 UI	USB Interrupt (USBINT) - R/WC. 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) has 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.

65.6.20 Interrupt Enable Register (USBC_*n*_USBINTR)

The interrupts to software are enabled with this register. An interrupt is generated when a bit is set and the corresponding interrupt source is active. The USB Status register (n_USBSTS) still shows interrupt sources even if they are disabled by the n_USBINTR register, allowing polling of interrupt events by the software.

Address: 218_4000h base + 148h offset + (512d × i), where i=0d to 3d



USBC_*n*_USBINTR field descriptions

Field	Description
31–26 -	This field is reserved. Reserved
25 TIE1	General Purpose Timer #1 Interrupt Enable When this bit is one and the TI1 bit in n_USBSTS register is a one the controller will issue an interrupt.
24 TIE0	General Purpose Timer #0 Interrupt Enable When this bit is one and the TI0 bit in n_USBSTS register is a one the controller will issue an interrupt.
23–20 -	This field is reserved. Reserved
19 UPIE	USB Host Periodic Interrupt Enable

Table continues on the next page...

USBC_*n*_USBINTR field descriptions (continued)

Field	Description
	When this bit is one, and the UPI bit in the n_USBSTS register is one, host controller will issue an interrupt at the next interrupt threshold.
18 UAIE	USB Host Asynchronous Interrupt Enable When this bit is one, and the UAI bit in the n_USBSTS register is one, host controller will issue an interrupt at the next interrupt threshold.
17 -	This field is reserved. Reserved
16 NAKE	NAK Interrupt Enable When this bit is one and the NAKI bit in n_USBSTS register is a one the controller will issue an interrupt.
15–11 -	These bits are reserved and should be set to zero.
10 ULPIE	ULPI Interrupt Enable When this bit is one and the UPLII bit in n_USBSTS register is a one the controller will issue an interrupt. This bit is usable only if the controller support UPLI interface mode.
9 -	This field is reserved. Reserved
8 SLE	Sleep Interrupt Enable When this bit is one and the SLI bit in n_n_USBSTS register is a one the controller will issue an interrupt. Only used in device operation mode.
7 SRE	SOF Received Interrupt Enable When this bit is one and the SRI bit in n_USBSTS register is a one the controller will issue an interrupt.
6 URE	USB Reset Interrupt Enable When this bit is one and the URI bit in n_USBSTS register is a one the controller will issue an interrupt. Only used in device operation mode.
5 AAE	Async Advance Interrupt Enable When this bit is one and the AAI bit in n_USBSTS register is a one the controller will issue an interrupt. Only used in host operation mode.
4 SEE	System Error Interrupt Enable When this bit is one and the SEI bit in n_USBSTS register is a one the controller will issue an interrupt. Only used in host operation mode.
3 FRE	Frame List Rollover Interrupt Enable When this bit is one and the FRI bit in n_USBSTS register is a one the controller will issue an interrupt. Only used in host operation mode.
2 PCE	Port Change Detect Interrupt Enable When this bit is one and the PCI bit in n_USBSTS register is a one the controller will issue an interrupt.
1 UEE	USB Error Interrupt Enable When this bit is one and the UEI bit in n_USBSTS register is a one the controller will issue an interrupt.
0 UE	USB Interrupt Enalbe When this bit is one and the UI bit in n_USBSTS register is a one the controller will issue an interrupt.

65.6.21 USB Frame Index (USBC_*n*_FRINDEX)

This register is used by the host controller to index the periodic frame list. The register updates every 125 microseconds (once each micro-frame). Bits [N: 3] are used to select a particular entry in the Periodic Frame List during periodic schedule execution. The number of bits used for the index depends on the size of the frame list as set by system software in the Frame List Size field in the n_USBCMD register.

This register must be written as a DWord. Byte writes produce-undefined results. This register cannot be written unless the Host Controller is in the 'Halted' state as indicated by the HCHalted bit. A write to this register while the Run/Stop hit is set to a one produces undefined results. Writes to this register also affect the SOF value.

In device mode this register is read only and, the device controller updates the FRINDEX [13:3] register from the frame number indicated by the SOF marker. Whenever a SOF is received by the USB bus, FRINDEX [13:3] will be checked against the SOF marker. If FRINDEX [13:3] is different from the SOF marker, FRINDEX [13:3] will be set to the SOF value and FRINDEX [2:0] will be set to zero (that is, SOF for 1 ms frame). If FRINDEX [13:3] is equal to the SOF value, FRINDEX [2:0] will be increment (that is, SOF for 125 us micro-frame.).

Address: 218_4000h base + 14Ch offset + (512d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																FRINDEX															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USBC_*n*_FRINDEX field descriptions

Field	Description
31–14 -	This field is reserved. Reserved
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. This means that 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 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 microframe. 000 (1024) 12 001 (512) 11

Table continues on the next page...

USBC_*n*_FRINDEX field descriptions (continued)

Field	Description
	010 (256) 10 011 (128) 9 100 (64) 8 101 (32) 7 110 (16) 6 111 (8) 5

65.6.22 Frame List Base Address (USBC_*n*_PERIODICLISTBASE)

Host Controller only

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 (USB_n_FRINDEX) to enable the Host Controller to step through the Periodic Frame List in sequence.

Address: 218_4000h base + 154h offset + (512d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BASEADR																Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USBC_*n*_PERIODICLISTBASE field descriptions

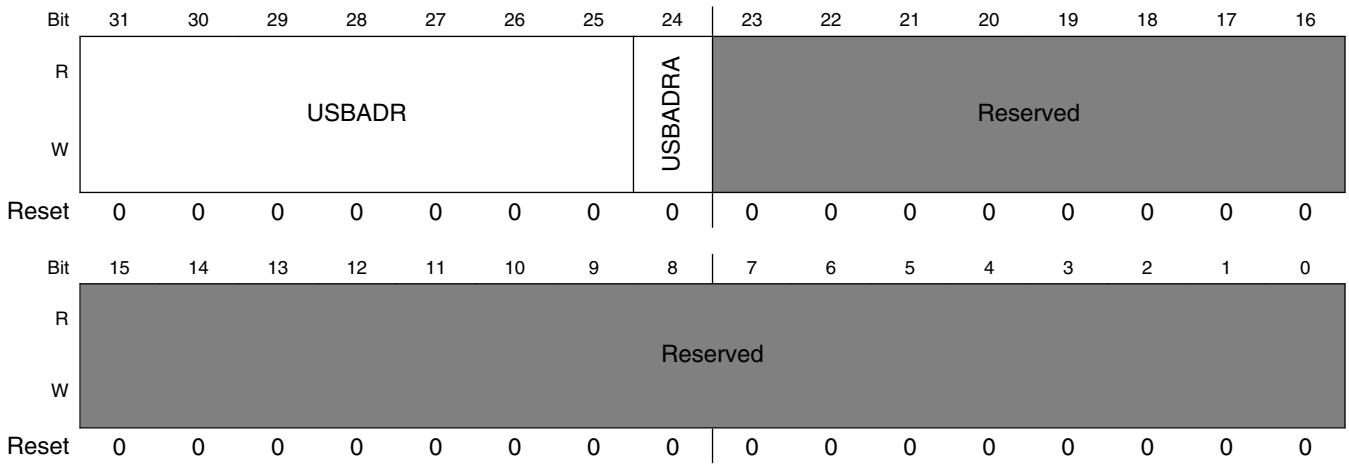
Field	Description
31–12 BASEADR	Base Address (Low). These bits correspond to memory address signals [31:12], respectively. Only used by the host controller.
-	This field is reserved. Reserved

65.6.23 Device Address (USBC_*n*_DEVICEADDR)

Device Controller only

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 descriptor.

Address: 218_4000h base + 154h offset + (512d × i), where i=0d to 0d



USBC_*n*_DEVICEADDR field descriptions

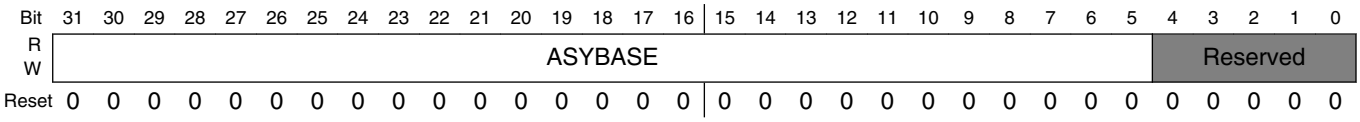
Field	Description
31–25 USBADR	Device Address. These bits correspond to the USB device address
24 USBADRA	Device Address Advance. Default=0. When this bit is '0', any writes to USBADR are instantaneous. When this bit is written to a '1' at the same time or before USBADR is written, the write to the USBADR 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.
-	This field is reserved. Reserved

65.6.24 Next Asynch. Address (USBC_*n*_ASYNCLISTADDR)

Host Controller only

This 32-bit register contains the address of the next asynchronous queue head to be executed by the host. Bits [4:0] of this register cannot be modified by the system software and will always return a zero when read.

Address: 218_4000h base + 158h offset + (512d × i), where i=0d to 3d



USBC_*n*_ASYNCLISTADDR field descriptions

Field	Description
31–5 ASYBASE	Link Pointer Low (LPL). These bits correspond to memory address signals [31:5], respectively. This field may only reference a Queue Head (QH). Only used by the host controller.
-	This field is reserved. Reserved

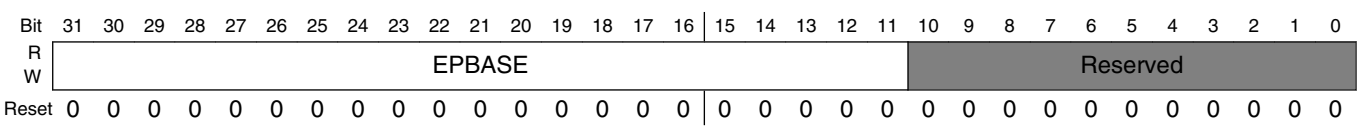
65.6.25 Endpoint List Address (USBC_*n*_ENDPTLISTADDR)

Device Controller only

In device mode, this register contains the address of the top of the endpoint list in system memory. Bits [10:0] of this register cannot be modified by the system software and will always return a zero when read.

The memory structure referenced by this physical memory pointer is assumed 64-byte.

Address: 218_4000h base + 158h offset + (512d × i), where i=0d to 0d



USBC_*n*_ENDPTLISTADDR field descriptions

Field	Description
31–11 EPBASE	Endpoint List Pointer(Low). These bits correspond to memory address signals [31:11], respectively. This field will reference a list of up to 32 Queue Head (OH) (that is, one queue head per endpoint & direction).
-	This field is reserved. Reserved

65.6.26 Programmable Burst Size (USBC_*n*_BURSTSIZE)

This register is used to control the burst size used during data movement on the AHB master interface. This register is ignored if AHBBRST bits in SBUSCFG register is non-zero value.

Address: 218_4000h base + 160h offset + (512d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																TXPBURST								RXPBURST							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USBC_*n*_BURSTSIZE field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16–8 TXPBURST	Programmable TX Burst Size. Default value is determined by TXBURST bits in n_HWTXBUF. This register represents the maximum length of a the burst in 32-bit words while moving data from system memory to the USB bus.
RXPBURST	Programmable RX Burst Size. Default value is determined by TXBURST bits in n_HWRXBUF. This register represents the maximum length of a the burst in 32-bit words while moving data from the USB bus to system memory.

65.6.27 TX FIFO Fill Tuning (USBC_*n*_TXFILLTUNING)

The fields in this register control performance tuning associated with how the host controller posts data to the TX latency FIFO before moving the data onto the USB bus. The specific areas of performance include the how much data to post into the FIFO and an estimate for how long that operation should take in the target system.

Definitions:

T_0 = Standard packet overhead

T_1 = Time to send data payload

T_{ff} = Time to fetch packet into TX FIFO up to specified level.

T_s = Total Packet Flight Time (send-only) packet

$T_s = T_0 + T_1$

T_p = Total Packet Time (fetch and send) packet

$T_p = T_{ff} + T_0 + T_1$

Upon discovery of a transmit (OUT/SETUP) packet in the data structures, host controller checks to ensure T_p remains before the end of the [micro]frame. If so it proceeds to pre-fill the TX FIFO. If at anytime during the pre-fill operation the time remaining the [micro]frame is < T_s then the packet attempt ceases and the packet is tried at a later time. Although this is not an error condition and the host controller will eventually recover, a mark will be made the scheduler health counter to note the occurrence of a "back-off" event. When a back-off event is detected, the partial packet fetched may need to be discarded from the latency buffer to make room for periodic traffic that will begin after the next SOF. Too many back-off events can waste bandwidth and power on the system bus and thus should be minimized (not necessarily eliminated). Back-offs can be minimized with use of the n_TSCHHEALTH (T_{ff}) described below.

NOTE

The reset value could vary from instance to instance. Please see the detail in bit field description and ignore reset value in summary table in this case!

Address: 218_4000h base + 164h offset + (512d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved										TXFIFOTHRES						Reserved			TXSCHHEALTH					TXSCHOH							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0

USBC_*n*_TXFILLTUNING field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–16 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 USB_n_USBMODE register is set. Default value is '00h' for OTG controller core, and '02h' for Host-only controller core.
15–13 -	This field is reserved. Reserved
12–8 TXSCHHEALTH	Scheduler Health Counter. (Read/Write To Clear) 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. Default value is '08h' for OTG controller core, and '00h' for Host-only controller core.

Table continues on the next page...

USBC_*n*_TXFILLTUNING field descriptions (continued)

Field	Description
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. Default value is '08h' for OTG controller core, and '00h' for Host-only controller core.

65.6.28 Endpoint NAK (USBC_*n*_ENDPTNAK)

Address: 218_4000h base + 178h offset + (512d × i), where i=0d to 0d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								EPTN								Reserved								EPRN							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USBC_*n*_ENDPTNAK field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 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. Bit [N] - Endpoint #[N], N is 0-7
15–8 -	This field is reserved. Reserved
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. Bit [N] - Endpoint #[N], N is 0-7

65.6.29 Endpoint NAK Enable (USBC_*n*_ENDPTNAKEN)

Address: 218_4000h base + 17Ch offset + (512d × i), where i=0d to 0d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								EPTNE								Reserved								EPRNE							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USBC_*n*_ENDPTNAKEN field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 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. Bit [N] - Endpoint #[N], N is 0-7
15–8 -	This field is reserved. Reserved
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. Bit [N] - Endpoint #[N], N is 0-7

65.6.30 Configure Flag Register (USBC_*n*_CONFIGFLAG)

Address: 218_4000h base + 180h offset + (512d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															CF
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

USBC_*n*_CONFIGFLAG field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 CF	Configure Flag Host software sets this bit as the last action in its process of configuring the Host Controller. This bit controls the default port-routing control logic. 0 Port routing control logic default-routes each port to an implementation dependent classic host controller. 1 Port routing control logic default-routes all ports to this host controller.

65.6.31 Port Status & Control (USBC_*n*_PORTSC1)

Host Controller

A host controller could implement one to eight port status and control registers. The number is determined by N_PORTs bits in HWSPARAMs register (please see Host Controller Structural Parameters (USBC_*n*_HCSPARAMS)). Software could read this parameter register to determine how many ports need service.

All controller cores are Single-Port Host, so there is only one port status and control register for each controller core.

This register is only reset by power on reset or controller core reset. The initial conditions of a port are:

- No device connected
- Port disabled

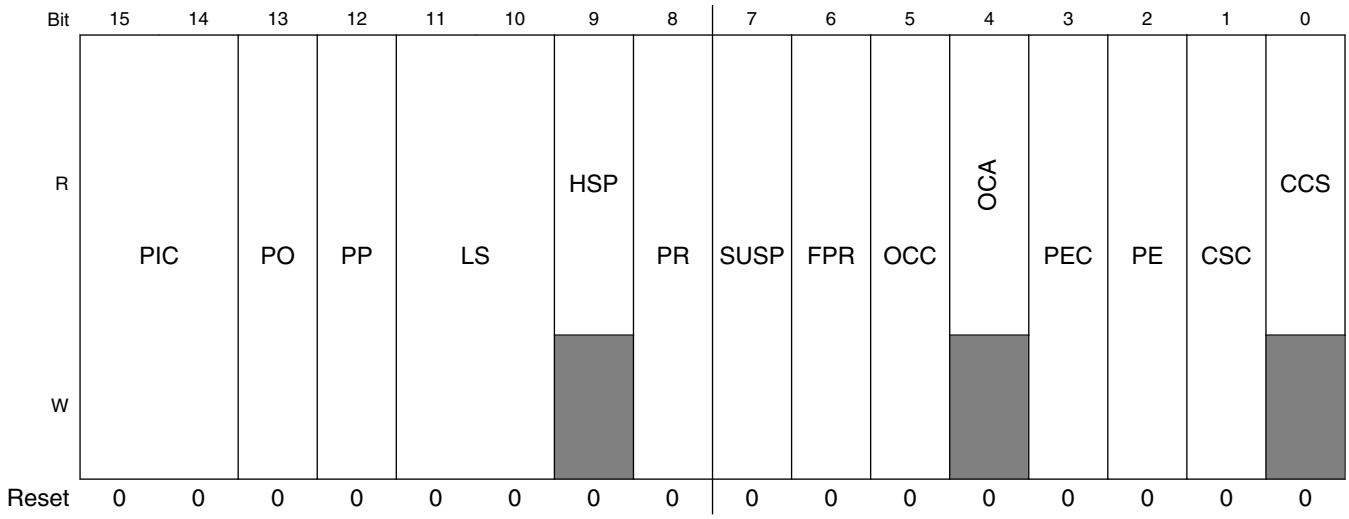
If the port supports power control, this state remains until port power is supplied (by software).

Device Controller

A device controller has only port register one (PORTSC1) and it does not support power control. Port control in device mode is only used for status port reset, suspend, and current connect status. It is also used to initiate test mode or force signaling and allows software to put the PHY into low power suspend mode and disable the PHY clock.

Address: 218_4000h base + 184h offset + (512d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PTS_1		STS	PTW	PSPD		PTS_2	PFSC	PHCD	WKOC	WKDC	WKCN	PTC			
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0



USBC_*n*_PORTSC1 field descriptions

Field	Description
31–30 PTS_1	Bit field {bit25, bit31, bit30}: "000b" UTMI/UTMI+ "001b" Reserved "010b" ULPI "011b" Serial/USB 1.1 PHY/IC-USB (FS Only) "100b" HSIC Parallel Transceiver Select (bit25, bit31, bi30). For OTG core, it is Read-Only. Reset value is 000b. For Host1/Host2/Host3 core, it is Read/Write. Reset value is 000b. **NOTE:** All USB port interface modes are listed in this field description, but not all are supported. For detail feature of each controller core, please see Features . The behaviour is unknown when unsupported interface mode is selected.
29 STS	Serial Transceiver Select - Read Only Serial Transceiver Select 1 Serial Interface Engine is selected 0 Parallel Interface signals is selected Serial Interface Engine can be used in combination with UTMI+/ULPI physical interface to provide FS/LS signaling instead of the parallel interface signals. When this bit is set '1b', serial interface engine will be used instead of parallel interface signals. This bit has no effect unless PTS bits is set to select UTMI+/ULPI interface. The Serial/USB1.1 PHY/IC-USB will use the serial interface engine for FS/LS signaling regardless of this bit value.
28 PTW	Parallel Transceiver Width This bit has no effect if serial interface engine is used. For OTG/Host1/Host2/Host3 core, it is Read-Only. Reset value is '1b'.

Table continues on the next page...

USBC_*n*_PORTSC1 field descriptions (continued)

Field	Description
	0 Select the 8-bit UTMI interface [60MHz] 1 Select the 16-bit UTMI interface [30MHz]
27–26 PSPD	Port Speed - Read Only. This register field indicates the speed at which the port is operating. 00 Full Speed 01 Low Speed 10 High Speed 11 Undefined
25 PTS_2	See description at bits 31-30
24 PFSC	Port Force Full Speed Connect - Read/Write. Default = 0b. When this bit is set to '1b', the port will be forced to only connect at Full Speed, It disables the chirp sequence that allows the port to identify itself as High Speed. 1 Forced to full speed 0 Normal operation
23 PHCD	PHY Low Power Suspend - Clock Disable (PLPSCD) - Read/Write. Default = 0b. When this bit is set to '1b', the PHY clock is disabled. Reading this bit will indicate the status of the PHY clock. **NOTE:** The PHY clock cannot be disabled if it is being used as the system clock. In device mode, The PHY can be put into Low Power Suspend when the device is not running (USBCMD Run/Stop=0b) or the host has signaled suspend (PORTSC1 SUSPEND=1b). PHY Low power suspend will be cleared automatically when the host initials 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 when the downstream device has been put into suspend mode or when no downstream device is connected. Low power suspend is completely under the control of software. 1 Disable PHY clock 0 Enable PHY clock
22 WKOC	Wake on Over-current Enable (WKOC_E) - Read/Write. Default = 0b. 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(*Port Status & Control (USBC_*n*_PORTSC1)*)* is zero.
21 WKDC	Wake on Disconnect Enable (WKDSCNNT_E) - Read/Write. Default=0b. 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(*Port Status & Control (USBC_*n*_PORTSC1)*)* is zero or in device mode.
20 WKCN	Wake on Connect Enable (WKCNNT_E) - Read/Write. Default=0b. 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(*Port Status & Control (USBC_*n*_PORTSC1)*)* is zero or in device mode.
19–16 PTC	Port Test Control - Read/Write. Default = 0000b. Refer to Port Test Mode for the operational model for using these test modes and the USB Specification Revision 2.0, Chapter 7 for details on each test mode.

Table continues on the next page...

USBC_*n*_PORTSC1 field descriptions (continued)

Field	Description
	The FORCE_ENABLE_FS and FORCE ENABLE_LS are extensions to the test mode support specified in the EHCI specification. Writing the PTC field to any of the FORCE_ENABLE_{HS/FS/LS} values will force the port into the connected and enabled state at the selected speed. Writing the PTC field back to TEST_MODE_DISABLE will allow the port state machines to progress normally from that point. **NOTE:** *Low speed operations are not supported as a peripheral device.* Any other value than zero indicates that the port is operating in test mode. Value Specific Test 0000 TEST_MODE_DISABLE 0001 J_STATE 0010 K_STATE 0011 SE0 (host) / NAK (device) 0100 Packet 0101 FORCE_ENABLE_HS 0110 FORCE_ENABLE_FS 0111 FORCE_ENABLE_LS 1000-1111 Reserved
15–14 PIC	Port Indicator Control - Read/Write. Default = Ob. Writing to this field has no effect if the P_INDICATOR bit in the HCSPARAMS register is a zero. Refer to the USB Specification Revision 2.0 for a description on how these bits are to be used. This field is zero if *Port Power* is zero. Bit Value Meaning 00 Port indicators are off 01 Amber 10 Green 11 Undefined
13 PO	Port Owner-Read/Write. Default = 0. This bit unconditionally goes to a 0 when the configured bit in the CONFIGFLAG register makes a 0 to 1 transition. This bit unconditionally goes to 1 whenever the Configured bit is zero System software uses this field to release ownership of the port to a selected host controller (in the event that the attached device is not a high-speed device). Software writes a one to this bit when the attached device is not a high-speed device. A one in this bit means that an internal companion controller owns and controls the port. Port owner handoff is not supported in all controller cores, therefore this bit will always be 0.
12 PP	Port Power (PP)-Read/Write or Read Only. 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 0 1b *Read Only - Host controller does not have port power control switches. Each port is hard-wired to power.* 1 1b/0b - *Read/Write. Host/OTG controller requires port power control switches. This bit represents the current setting of the switch (0=off, 1=on). When power is not available on a port (that is, PP equals a 0), the port is non-functional and will not report attaches, detaches, etc.*

Table continues on the next page...

USBC_*n*_PORTSC1 field descriptions (continued)

Field	Description
	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 transitional by the host controller driver from a one to a zero (removing power from the port). This feature is implemented in all controller cores (PPC = 1).
11–10 LS	Line Status-Read Only. These bits reflect the current logical levels of the D+ (bit 11) and D- (bit 10) signal lines. In host mode, the use of linestate 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 linestate by the device controller driver is not necessary. The encoding of the bits are: Bits [11:10] Meaning 00 SE0 10 J-state 01 K-state 11 Undefined
9 HSP	High-Speed Port - Read Only. Default = 0b. 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(bit 27, 26) but remained for compatibility.
8 PR	Port Reset - Read/Write or Read Only. Default = 0b. In Host Mode: Read/Write. 1=Port is in Reset. 0=Port is not in Reset. Default 0. 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. This field is zero if *Port Power(*Port Status & Control (USBC_*n*_PORTSC1)*)* is zero.
7 SUSP	Suspend - Read/Write or Read Only. Default = 0b. 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

Table continues on the next page...

USBC_*n*_PORTSC1 field descriptions (continued)

Field	Description
	The host controller will unconditionally set this bit to zero when software sets the *Force Port Resume* bit to zero. The host controller ignores a write of zero to this bit. If host software sets this bit to a one when the port is not enabled (that is, *Port enabled* bit is a zero) the results are undefined. This field is zero if *Port Power(*Port Status & Control (USBC_*n*_PORTSC1)*)* is zero in host mode. In Device Mode: Read Only. In device mode this bit is a read only status bit.
6 FPR	Force Port Resume -Read/Write. 1= Resume detected/driven on port. 0=No resume (K-state) detected/ driven on port. Default = 0. 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 will remain 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 clear the bit the port control state switches to HS or FS idle. This field is zero if *Port Power(*Port Status & Control (USBC_*n*_PORTSC1)*)* 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 wil be cleared because a K-to-J transition detected, the *Port Change Detect* bit in the USBSTS register is also set to one.
5 OCC	Over-current Change-R/WC. Default=0. This bit is set '1b' by hardware when there is a change to Over-current Active. Software can clear this bit by writing a one to this bit position.
4 OCA	Over-current Active-Read Only. Default 0. This bit will automatically transition from one to zero when the over current condition is removed. 1 This port currently has an over-current condition 0 This port does not have an over-current condition.
3 PEC	Port Enable/Disable Change-R/WC. 1=Port enabled/disabled status has changed. 0=No change. Default = 0. In Host Mode: For the root hub, this bit is 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(*Port Status & Control (USBC_*n*_PORTSC1)*)* is zero. In Device mode:

Table continues on the next page...

USBC_*n*_PORTSC1 field descriptions (continued)

Field	Description
	The device port is always enabled, so this bit is always '0b'.
2 PE	Port Enabled/Disabled-Read/Write. 1=Enable. 0=Disable. Default 0. 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(*Port Status & Control (USBC_*n*_PORTSC1)*)* is zero in host mode. In Device Mode: The device port is always enabled, so this bit is always '1b'.
1 CSC	Connect Status Change-R/WC. 1 =Change in Current Connect Status. 0=No change. Default 0. 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 (that is, the bit will remain set). Software clears this bit by writing a one to it. This field is zero if *Port Power(*Port Status & Control (USBC_*n*_PORTSC1)*)* is zero in host mode. In Device Mode: This bit is undefined in device controller mode.
0 CCS	Current Connect Status-Read Only. In Host Mode: 1=Device is present on port. 0=No device is present. Default = 0. 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(*Port Status & Control (USBC_*n*_PORTSC1)*)* is zero in host mode. In Device Mode: 1=Attached. 0=Not Attached. Default=0. 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.

65.6.32 On-The-Go Status & control (USBC_*n*_OTGSC)

This register is availabe only in OTG controller core. It has four sections:

- OTG Interrupt enables (Read/Write)
- OTG Interrupt status (Read/Write to Clear)

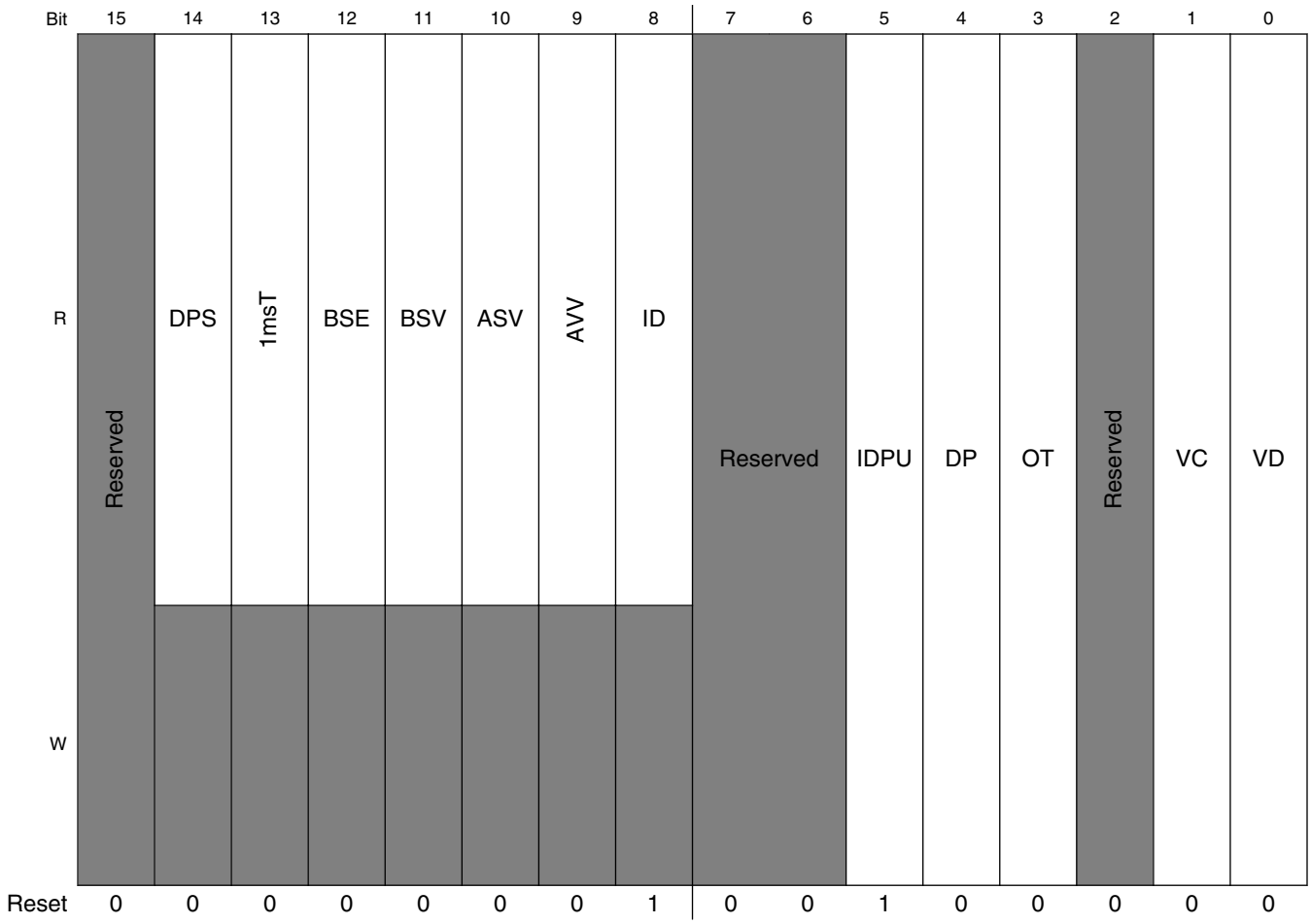
- OTG Status inputs (Read Only)
- OTG Controls (Read/Write)

The status inputs are debounced using a 1 ms time constant. Values on the status inputs that do not persist for more than 1 ms does not cause an update of the status input register, or cause an OTG interrupt.

See also USB Device Mode (USBC_*n*_USBMODE) register.

Address: 218_4000h base + 1A4h offset + (512d × i), where i=0d to 0d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved	DPIE	1msE	BSEIE	BSVIE	ASVIE	AVVIE	IDIE	Reserved	DPIS	1msS	BSEIS	BSVIS	ASVIS	AVVIS	IDIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



USBC_*n*_OTGSC field descriptions

Field	Description
31 -	This field is reserved. Reserved
30 DPIE	Data Pulse Interrupt Enable
29 1msE	1 millisecond timer Interrupt Enable - Read/Write
28 BSEIE	B Session End Interrupt Enable - Read/Write. Setting this bit enables the B session end interrupt.
27 BSVIE	B Session Valid Interrupt Enable - Read/Write. Setting this bit enables the B session valid interrupt.
26 ASVIE	A Session Valid Interrupt Enable - Read/Write. Setting this bit enables the A session valid interrupt.
25 AVVIE	A VBus Valid Interrupt Enable - Read/Write. Setting this bit enables the A VBus valid interrupt.
24 IDIE	USB ID Interrupt Enable - Read/Write. Setting this bit enables the USB ID interrupt.

Table continues on the next page...

USBC_*n*_OTGSC field descriptions (continued)

Field	Description
23 -	This field is reserved. Reserved
22 DPIS	Data Pulse Interrupt Status - Read/Write to Clear. 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 PORTSC1(0)[PP] = 0. Software must write a one to clear this bit.
21 1msS	1 millisecond timer Interrupt Status - Read/Write to Clear. This bit is set once every millisecond. Software must write a one to clear this bit.
20 BSEIS	B Session End Interrupt Status - Read/Write to Clear. This bit is set when VBus has fallen below the B session end threshold. Software must write a one to clear this bit
19 BSVIS	B Session Valid Interrupt Status - Read/Write to Clear. This bit is set when VBus has either risen above or fallen below the B session valid threshold. Software must write a one to clear this bit.
18 ASVIS	A Session Valid Interrupt Status - Read/Write to Clear. This bit is set when VBus has either risen above or fallen below the A session valid threshold. Software must write a one to clear this bit.
17 AVVIS	A VBus Valid Interrupt Status - Read/Write to Clear. This bit is set when VBus has either risen above or fallen below the VBus valid threshold on an A device. Software must write a one to clear this bit.
16 IDIS	USB ID Interrupt Status - Read/Write. This bit is set when a change on the ID input has been detected. Software must write a one to clear this bit.
15 -	This field is reserved. Reserved
14 DPS	Data Bus Pulsing Status - Read Only. A '1' indicates data bus pulsing is being detected on the port.
13 1msT	1 millisecond timer toggle - Read Only. This bit toggles once per millisecond.
12 BSE	B Session End - Read Only. Indicates VBus is below the B session end threshold.
11 BSV	B Session Valid - Read Only. Indicates VBus is above the B session valid threshold.
10 ASV	A Session Valid - Read Only. Indicates VBus is above the A session valid threshold.
9 AVV	A VBus Valid - Read Only. Indicates VBus is above the A VBus valid threshold.
8 ID	USB ID - Read Only.

Table continues on the next page...

USBC_*n*_OTGSC field descriptions (continued)

Field	Description
	0 = A device, 1 = B device
7–6 -	This field is reserved. Reserved
5 IDPU	ID Pullup - Read/Write This bit provide control over the ID pull-up resister; 0 = off, 1 = on [default]. When this bit is 0, the ID input will not be sampled.
4 DP	Data Pulsing - Read/Write. Setting this bit causes the pullup on DP to be asserted for data pulsing during SRP.
3 OT	OTG Termination - Read/Write. This bit must be set when the OTG device is in device mode, this controls the pulldown on DM.
2 -	This field is reserved. Reserved
1 VC	VBUS Charge - Read/Write. Setting this bit causes the VBus line to be charged. This is used for VBus pulsing during SRP.
0 VD	VBUS_Discharge - Read/Write. Setting this bit causes VBus to discharge through a resistor.

65.6.33 USB Device Mode (USBC_*n*_USBMODE)

Address: 218_4000h base + 1A8h offset + (512d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved	Reserved										SDIS	SLOM	ES	CM	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USBC_*n*_USBMODE field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
15 -	This field is reserved. Reserved
14–5 -	This field is reserved. Reserved
4 SDIS	Stream Disable Mode. (0 - Inactive [default]; 1 - Active) Device Mode: Setting to a '1' disables double priming on both RX and TX for low bandwidth systems. This mode ensures that when the RX and TX buffers are sufficient to contain an entire packet that the standard double buffering scheme is disabled to prevent overruns/underruns in bandwidth limited systems. Note: In High Speed Mode, all packets received are responded to with a NYET handshake when stream disable is active. Host Mode: Setting to a '1' ensures that overruns/underruns of the latency FIFO are eliminated for low bandwidth systems where the RX and TX buffers are sufficient to contain the entire packet. Enabling stream disable also has the effect of ensuring the TX latency is filled to capacity before the packet is launched onto the USB. **NOTE:** Time duration to pre-fill the FIFO becomes significant when stream disable is active. See TX FIFO Fill Tuning (USBC_*n*_TXFILLTUNING) and TXTTFILLTUNING [MPH Only] to characterize the adjustments needed for the scheduler when using this feature. **NOTE:** The use of this feature substantially limits of the overall USB performance that can be achieved.
3 SLOM	Setup Lockout Mode. In device mode, this bit controls behavior of the setup lock mechanism. See Control Endpoint Operation Model . 0 Setup Lockouts On (default); 1 Setup Lockouts Off (DCD requires use of Setup Data Buffer Tripwire in USB Command Register (USBC_*n*_USBCMD) .
2 ES	Endian Select - Read/Write. This bit can change the byte alignment of the transfer buffers to match the host microprocessor. The bit fields in the microprocessor interface and the data structures are unaffected by the value of this bit because they are based upon the 32-bit word. Bit Meaning 0 Little Endian [Default] 1 Big Endian
CM	Controller Mode - R/WO. Controller mode is defaulted to the proper mode for host only and device only implementations. For those designs that contain both host & device capability, the controller defaults to an idle state and needs to be initialized to the desired operating mode after reset. For combination host/device controllers, 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. For OTG controller core, reset value is '00b'. For Host-only controller core, reset value is '11b'. 00 Idle [Default for combination host/device] 01 Reserved 10 Device Controller [Default for device only controller] 11 Host Controller [Default for host only controller]

65.6.34 Endpoint Setup Status (USBC_*n*_ENDPTSETUPSTAT)

Address: 218_4000h base + 1ACh offset + (512d × i), where i=0d to 0d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																ENDPTSETUPSTAT																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USBC_*n*_ENDPTSETUPSTAT field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
ENDPTSETUPSTAT	Setup Endpoint Status. For every setup transaction that is received, a corresponding bit in this register is set to one. Software must clear or acknowledge the setup transfer by writing a one to a respective bit 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 our mechanism is engaged. See Managing Endpoints in the Device Operational Model. This register is only used in device mode.

65.6.35 Endpoint Prime (USBC_*n*_ENDPTPRIME)

This register is only used in device mode.

When software sets the prime bit for a given endpoint, the device controller loads the transfer descriptor, pointed to by the queue head, such that the endpoint is ready to transmit or receive when the host sends a request (IN/OUT token). The endpoint will NAK all requests from the host until the endpoint is primed. The controller will automatically re-prime the endpoint with a new transfer descriptor when one is found via the next_dtd pointer of the current transfer descriptor. Hence, the prime bit must only be set by software when a descriptor is added to the queue head.

Address: 218_4000h base + 1B0h offset + (512d × i), where i=0d to 0d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								PETB								Reserved								PERB							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USBC_*n*_ENDPTPRIME field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 PETB	Prime Endpoint Transmit Buffer - R/WS. For each endpoint a corresponding bit is used to request that a buffer is prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction.

Table continues on the next page...

USBC_*n*_ENDPTPRIME field descriptions (continued)

Field	Description
	Software should write a one to the corresponding bit when posting a new transfer descriptor to an endpoint queue head. Hardware automatically uses this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware clears this bit when the associated endpoint(s) is (are) successfully primed. **NOTE:** These bits are momentarily set by hardware during hardware re-priming operations when a dTD is retired, and the dQH is updated. PETB[N] - Endpoint #N, N is in 0..7
15–8 -	This field is reserved. Reserved
PERB	Prime Endpoint Receive Buffer - R/WS. For each endpoint, a corresponding bit is used to request a buffer prepare for a receive operation for when a USB host initiates a USB OUT transaction. Software should write a one to the corresponding bit whenever posting a new transfer descriptor to an endpoint queue head. Hardware automatically uses this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware clears this bit when the associated endpoint(s) is (are) successfully primed. **NOTE:** These bits are momentarily set by hardware during hardware re-priming operations when a dTD is retired, and the dQH is updated. PERB[N] - Endpoint #N, N is in 0..7

65.6.36 Endpoint Flush (USBC_*n*_ENDPTFLUSH)

This register is only used in device mode.

Address: 218_4000h base + 1B4h offset + (512d × i), where i=0d to 0d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								FETB								Reserved								FERB							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USBC_*n*_ENDPTFLUSH field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 FETB	Flush Endpoint Transmit Buffer - R/WS. Writing one to a bit(s) in this register causes the associated endpoint(s) to clear any primed buffers. If a packet is in progress for one of the associated endpoints, then that transfer continues until completion. Hardware clears this register after the endpoint flush operation is successful. FETB[N] - Endpoint #N, N is in 0..7
15–8 -	This field is reserved. Reserved

Table continues on the next page...

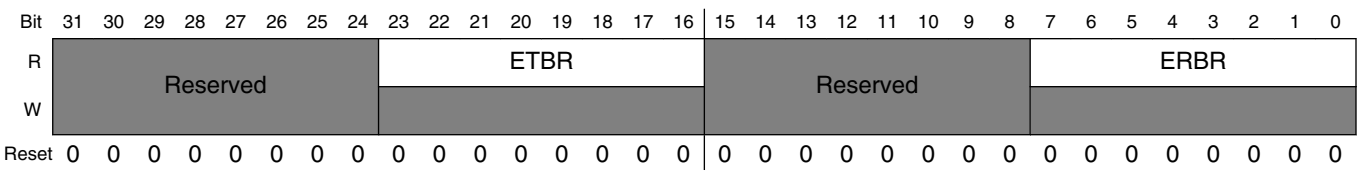
USBC_*n*_ENDPTFLUSH field descriptions (continued)

Field	Description
FERB	Flush Endpoint Receive Buffer - R/WS. Writing one to a bit(s) causes the assocUOGiated endpoint(s) to clear any primed buffers. If a packet is in progress for one of the associated endpoints, then that transfer continues until completion. Hardware clears this register after the endpoint flush operation is successful. FERB[N] - Endpoint #N, N is in 0..7\

65.6.37 Endpoint Status (USBC_*n*_ENDPTSTAT)

This register is only used in device mode.

Address: 218_4000h base + 1B8h offset + (512d × i), where i=0d to 0d



USBC_*n*_ENDPTSTAT field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 ETBR	Endpoint Transmit Buffer Ready -- Read Only. One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There is always a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. **NOTE:** These bits are momentarily cleared by hardware during hardware endpoint re-priming operations when a dTD is retired, and the dQH is updated. ETBR[N] - Endpoint #N, N is in 0..7
15–8 -	This field is reserved. Reserved
ERBR	Endpoint Receive Buffer Ready -- Read Only. One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPRIME register. There is always a delay between setting a bit in the ENDPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. **NOTE:** These bits are momentarily cleared by hardware during hardware endpoint re-priming operations when a dTD is retired, and the dQH is updated. ERBR[N] - Endpoint #N, N is in 0..7

65.6.38 Endpoint Complete (USBC_*n*_ENDPTCOMPLETE)

This register is only used in device mode.

Address: 218_4000h base + 1BCh offset + (512d × i), where i=0d to 0d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								ETCE								Reserved								ERCE							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

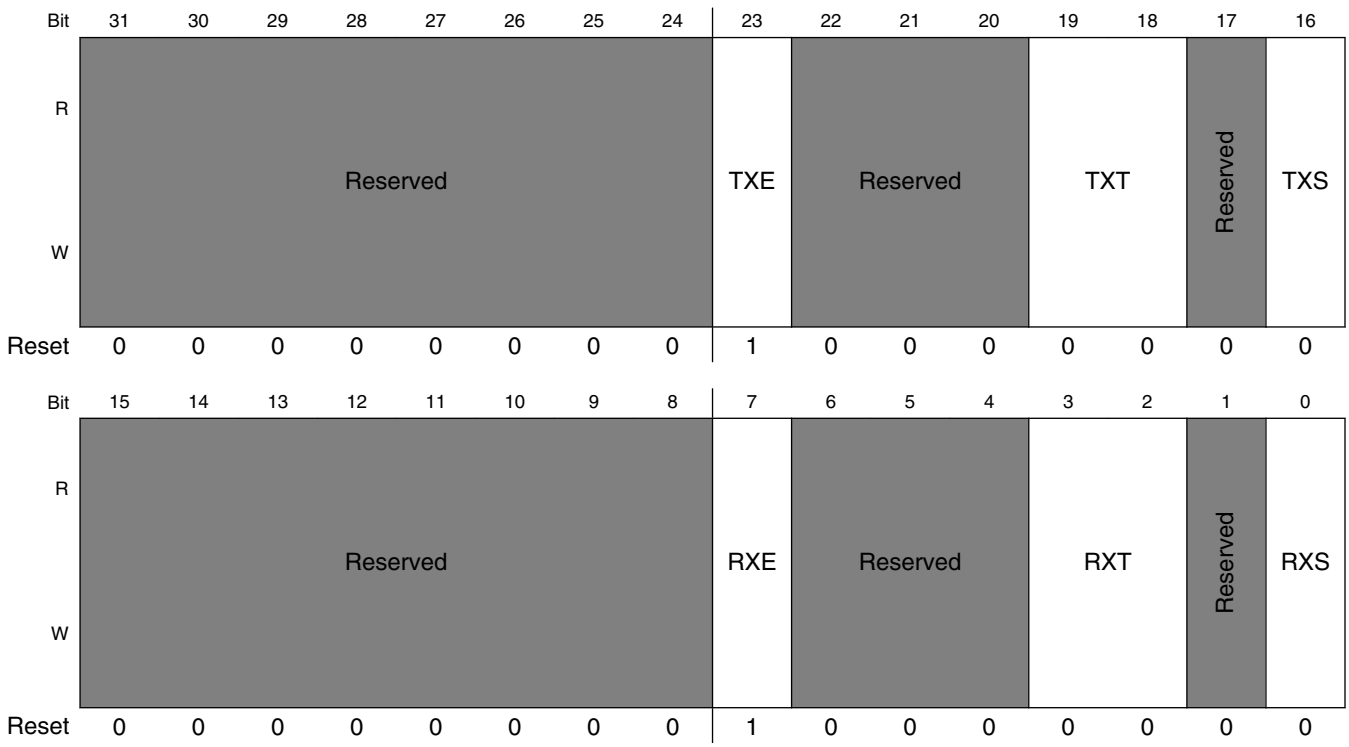
USBC_*n*_ENDPTCOMPLETE field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 ETCE	Endpoint Transmit Complete Event - R/WC. 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 is set simultaneously with the *USBINT* . Writing one clears the corresponding bit in this register. ETCE[N] - Endpoint #N, N is in 0..7
15–8 -	This field is reserved. Reserved
ERCE	Endpoint Receive Complete Event - RW/C. 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 is set simultaneously with the *USBINT* . Writing one clears the corresponding bit in this register. ERCE[N] - Endpoint #N, N is in 0..7

65.6.39 Endpoint Control0 (USBC_*n*_ENDPTCTRL0)

Every Device implements Endpoint 0 as a control endpoint.

Address: 218_4000h base + 1C0h offset + (512d × i), where i=0d to 0d



USBC_*n*_ENDPTCTRL0 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23 TXE	TX Endpoint Enable 1 Enabled Endpoint0 is always enabled.
22–20 -	This field is reserved. Reserved
19–18 TXT	TX Endpoint Type - Read/Write 00 - Control Endpoint0 is fixed as a Control End Point.
17 -	This field is reserved. Reserved
16 TXS	TX Endpoint Stall - Read/Write

Table continues on the next page...

USBC_*n*_ENDPTCTRL0 field descriptions (continued)

Field	Description
	0 End Point OK [Default] 1 End Point Stalled Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It continues returning STALL until the bit is cleared by software or it is automatically cleared upon receipt of a new SETUP request.
15–8 -	This field is reserved. Reserved
7 RXE	RX Endpoint Enable 1 Enabled Endpoint0 is always enabled.
6–4 -	This field is reserved. Reserved
3–2 RXT	RX Endpoint Type - Read/Write 00 Control Endpoint0 is fixed as a Control End Point.
1 -	This field is reserved. Reserved
0 RXS	RX Endpoint Stall - Read/Write 0 End Point OK. [Default] 1 End Point Stalled Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It continues returning STALL until the bit is cleared by software or it is automatically cleared upon receipt of a new SETUP request.

65.6.40 Endpoint Control 1 (USBC_*n*_ENDPTCTRL1)

This is endpoint control register for endpoint 1 in device operation mode.

NOTE

If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (that is Bulk-type). leaving an unconfigured endpoint control causes undefined behavior for the data pid tracking on the active endpoint/direction.

Address: 218_4000h base + 1C4h offset + (512d × i), where i=0d to 0d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								TXE	TXR	TXI	Reserved	TXT		TXD	TXS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								RXE	RXR	RXI	Reserved	RXT		RXD	RXS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USBC_*n*_ENDPTCTRL1 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23 TXE	TX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
22 TXR	TX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device.
21 TXI	TX Data Toggle Inhibit 0 PID Sequencing Enabled. [Default] 1 PID Sequencing Disabled. This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.
20 -	This field is reserved. Reserved
19–18 TXT	TX Endpoint Type - Read/Write 00 Control 01 Isochronous

Table continues on the next page...

USBC_*n*_ENDPTCTRL1 field descriptions (continued)

Field	Description
	10 Bulk 11 Interrupt
17 TXD	TX Endpoint Data Source - Read/Write 0 Dual Port Memory Buffer/DMA Engine [DEFAULT] Should always be written as 0.
16 TXS	TX Endpoint Stall - Read/Write 0 End Point OK 1 End Point Stalled This bit is set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It is 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 continues to returning STALL until this bit is either cleared by software or automatically cleared as above. **NOTE:** For CONTROL type endpoint, there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. Take care that the STALL bit is not set immediately after writing a '1' to it. Please follow this procedure: continually write this STALL bit until it is set or until a new setup has ben received by checking the associated ENDPTSETUPSTAT bit.
15–8 -	This field is reserved. Reserved
7 RXE	RX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
6 RXR	RX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device.
5 RXI	RX Data Toggle Inhibit 0 Disabled [Default] 1 Enabled This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.
4 -	This field is reserved. Reserved.
3–2 RXT	RX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Reserved

Table continues on the next page...

USBC_*n*_ENDPTCTRL1 field descriptions (continued)

Field	Description
1 RXD	RX Endpoint Data Sink - Read/Write - TBD 0 Dual Port Memory Buffer/DMA Engine [Default] Should always be written as zero.
0 RXS	RX Endpoint Stall - Read/Write 0 End Point OK. [Default] 1 End Point Stalled This bit is set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It is 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 continues to returning STALL until this bit is either cleared by software or automatically cleared as above.

65.6.41 Endpoint Control 2 (USBC_*n*_ENDPTCTRL2)

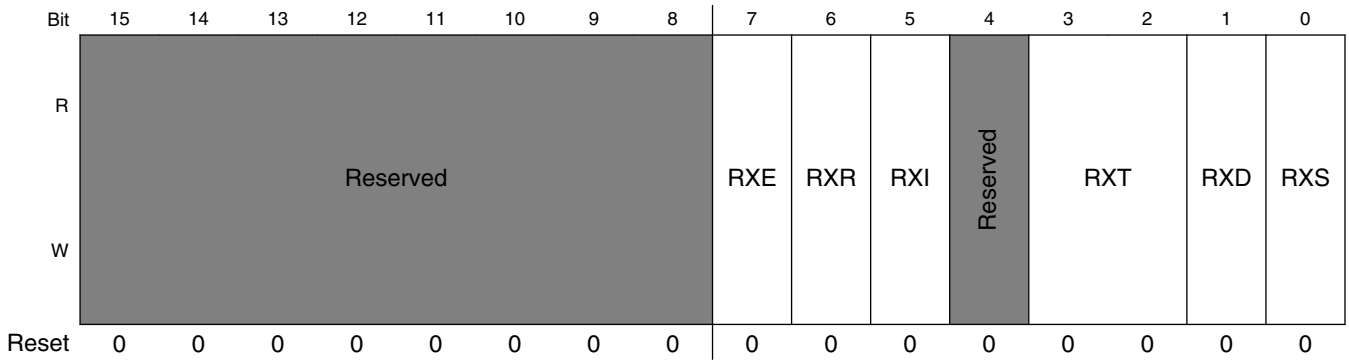
This is endpoint control register for endpoint 2 in device operation mode.

NOTE

If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (that is Bulk-type). leaving an unconfigured endpoint control causes undefined behavior for the data pid tracking on the active endpoint/direction.

Address: 218_4000h base + 1C8h offset + (512d × i), where i=0d to 0d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								TXE	TXR	TXI	Reserved	TXT		TXD	TXS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



USBC_*n*_ENDPTCTRL2 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23 TXE	TX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
22 TXR	TX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device.
21 TXI	TX Data Toggle Inhibit 0 PID Sequencing Enabled. [Default] 1 PID Sequencing Disabled. This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.
20 -	This field is reserved. Reserved
19–18 TXT	TX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt
17 TXD	TX Endpoint Data Source - Read/Write 0 Dual Port Memory Buffer/DMA Engine [DEFAULT] Should always be written as 0.
16 TXS	TX Endpoint Stall - Read/Write 0 End Point OK 1 End Point Stalled

Table continues on the next page...

USBC_*n*_ENDPTCTRL2 field descriptions (continued)

Field	Description
	This bit is set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It is 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 continues to returning STALL until this bit is either cleared by software or automatically cleared as above. **NOTE:** For CONTROL type endpoint, there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. Take care that the STALL bit is not set immediately after writing a '1' to it. Please follow this procedure: continually write this STALL bit until it is set or until a new setup has ben received by checking the associated ENDPTSETUPSTAT bit.
15–8 -	This field is reserved. Reserved
7 RXE	RX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
6 RXR	RX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device.
5 RXI	RX Data Toggle Inhibit 0 Disabled [Default] 1 Enabled This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.
4 -	This field is reserved. Reserved.
3–2 RXT	RX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Reserved
1 RXD	RX Endpoint Data Sink - Read/Write - TBD 0 Dual Port Memory Buffer/DMA Engine [Default] Should always be written as zero.
0 RXS	RX Endpoint Stall - Read/Write 0 End Point OK. [Default] 1 End Point Stalled This bit is set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It is cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint,

Table continues on the next page...

USBC_*n*_ENDPTCTRL2 field descriptions (continued)

Field	Description
	Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It continues to returning STALL until this bit is either cleared by software or automatically cleared as above.

65.6.42 Endpoint Control 3 (USBC_*n*_ENDPTCTRL3)

This is endpoint control register for endpoint 3 in device operation mode.

NOTE

If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (that is Bulk-type). leaving an unconfigured endpoint control causes undefined behavior for the data pid tracking on the active endpoint/direction.

Address: 218_4000h base + 1CCh offset + (512d × i), where i=0d to 0d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								TXE	TXR	TXI	Reserved	TXT		TXD	TXS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								RXE	RXR	RXI	Reserved	RXT		RXD	RXS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USBC_*n*_ENDPTCTRL3 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved

Table continues on the next page...

USBC_*n*_ENDPTCTRL3 field descriptions (continued)

Field	Description
23 TXE	TX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
22 TXR	TX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device.
21 TXI	TX Data Toggle Inhibit 0 PID Sequencing Enabled. [Default] 1 PID Sequencing Disabled. This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.
20 -	This field is reserved. Reserved
19–18 TXT	TX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt
17 TXD	TX Endpoint Data Source - Read/Write 0 Dual Port Memory Buffer/DMA Engine [DEFAULT] Should always be written as 0.
16 TXS	TX Endpoint Stall - Read/Write 0 End Point OK 1 End Point Stalled This bit is set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It is 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 continues to returning STALL until this bit is either cleared by software or automatically cleared as above. **NOTE:** For CONTROL type endpoint, there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. Take care that the STALL bit is not set immediately after writing a '1' to it. Please follow this procedure: continually write this STALL bit until it is set or until a new setup has ben received by checking the associated ENDPTSETUPSTAT bit.
15–8 -	This field is reserved. Reserved
7 RXE	RX Endpoint Enable 0 Disabled [Default]

Table continues on the next page...

USBC_*n*_ENDPTCTRL3 field descriptions (continued)

Field	Description
	1 Enabled An Endpoint should be enabled only after it has been configured.
6 RXR	RX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device.
5 RXI	RX Data Toggle Inhibit 0 Disabled [Default] 1 Enabled This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.
4 -	This field is reserved. Reserved.
3–2 RXT	RX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Reserved
1 RXD	RX Endpoint Data Sink - Read/Write - TBD 0 Dual Port Memory Buffer/DMA Engine [Default] Should always be written as zero.
0 RXS	RX Endpoint Stall - Read/Write 0 End Point OK. [Default] 1 End Point Stalled This bit is set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It is 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 continues to returning STALL until this bit is either cleared by software or automatically cleared as above.

65.6.43 Endpoint Control 4 (USBC_*n*_ENDPTCTRL4)

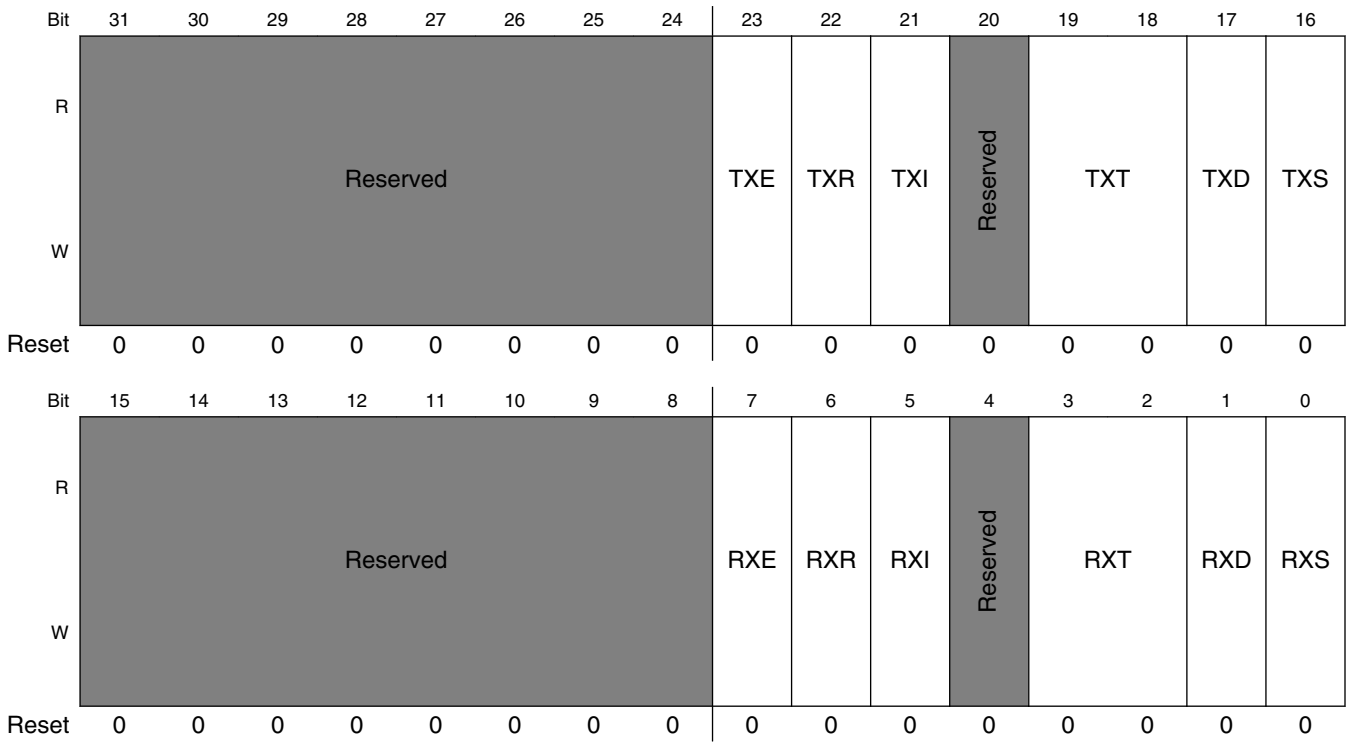
This is endpoint control register for endpoint 4 in device operation mode.

NOTE

If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (that is Bulk-type). leaving an unconfigured endpoint control

causes undefined behavior for the data pid tracking on the active endpoint/direction.

Address: 218_4000h base + 1D0h offset + (512d × i), where i=0d to 0d



USBC_*n*_ENDPTCTRL4 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23 TXE	TX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
22 TXR	TX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device.
21 TXI	TX Data Toggle Inhibit 0 PID Sequencing Enabled. [Default] 1 PID Sequencing Disabled. This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.
20 -	This field is reserved. Reserved

Table continues on the next page...

USBC_*n*_ENDPTCTRL4 field descriptions (continued)

Field	Description
19–18 TXT	TX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt
17 TXD	TX Endpoint Data Source - Read/Write 0 Dual Port Memory Buffer/DMA Engine [DEFAULT] Should always be written as 0.
16 TXS	TX Endpoint Stall - Read/Write 0 End Point OK 1 End Point Stalled This bit is set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It is 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 continues to returning STALL until this bit is either cleared by software or automatically cleared as above. **NOTE:** For CONTROL type endpoint, there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. Take care that the STALL bit is not set immediately after writing a '1' to it. Please follow this procedure: continually write this STALL bit until it is set or until a new setup has ben received by checking the associated ENDPTSETUPSTAT bit.
15–8 -	This field is reserved. Reserved
7 RXE	RX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
6 RXR	RX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device.
5 RXI	RX Data Toggle Inhibit 0 Disabled [Default] 1 Enabled This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.
4 -	This field is reserved. Reserved.
3–2 RXT	RX Endpoint Type - Read/Write 00 Control

Table continues on the next page...

USBC_*n*_ENDPTCTRL4 field descriptions (continued)

Field	Description
	01 Isochronous 10 Bulk 11 Reserved
1 RXD	RX Endpoint Data Sink - Read/Write - TBD 0 Dual Port Memory Buffer/DMA Engine [Default] Should always be written as zero.
0 RXS	RX Endpoint Stall - Read/Write 0 End Point OK. [Default] 1 End Point Stalled This bit is set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It is 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 continues to returning STALL until this bit is either cleared by software or automatically cleared as above.

65.6.44 Endpoint Control 5 (USBC_*n*_ENDPTCTRL5)

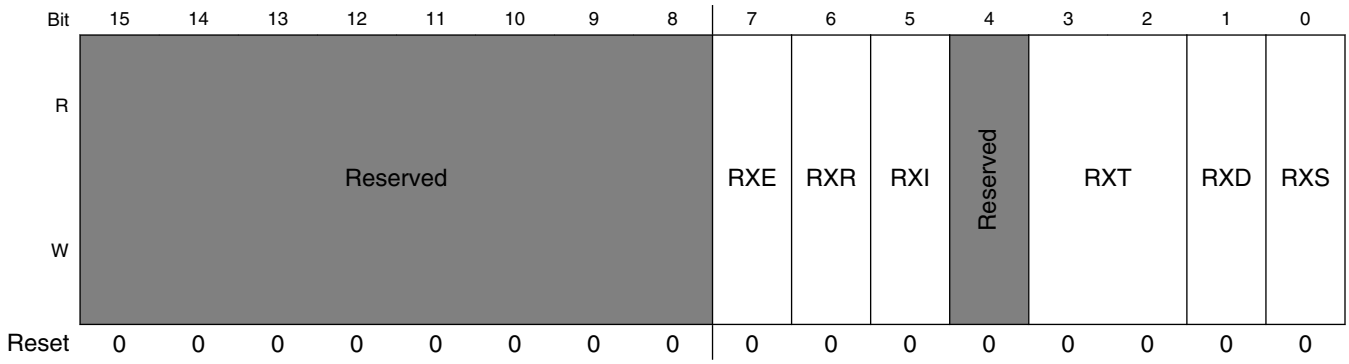
This is endpoint control register for endpoint 5 in device operation mode.

NOTE

If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (that is Bulk-type). leaving an unconfigured endpoint control causes undefined behavior for the data pid tracking on the active endpoint/direction.

Address: 218_4000h base + 1D4h offset + (512d × i), where i=0d to 0d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								TXE	TXR	TXI	Reserved	TXT		TXD	TXS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



USBC_*n*_ENDPTCTRL5 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23 TXE	TX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
22 TXR	TX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device.
21 TXI	TX Data Toggle Inhibit 0 PID Sequencing Enabled. [Default] 1 PID Sequencing Disabled. This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.
20 -	This field is reserved. Reserved
19–18 TXT	TX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt
17 TXD	TX Endpoint Data Source - Read/Write 0 Dual Port Memory Buffer/DMA Engine [DEFAULT] Should always be written as 0.
16 TXS	TX Endpoint Stall - Read/Write 0 End Point OK 1 End Point Stalled

Table continues on the next page...

USBC_*n*_ENDPTCTRL5 field descriptions (continued)

Field	Description
	This bit is set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It is 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 continues to returning STALL until this bit is either cleared by software or automatically cleared as above. **NOTE:** For CONTROL type endpoint, there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. Take care that the STALL bit is not set immediately after writing a '1' to it. Please follow this procedure: continually write this STALL bit until it is set or until a new setup has ben received by checking the associated ENDPTSETUPSTAT bit.
15–8 -	This field is reserved. Reserved
7 RXE	RX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
6 RXR	RX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device.
5 RXI	RX Data Toggle Inhibit 0 Disabled [Default] 1 Enabled This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.
4 -	This field is reserved. Reserved.
3–2 RXT	RX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Reserved
1 RXD	RX Endpoint Data Sink - Read/Write - TBD 0 Dual Port Memory Buffer/DMA Engine [Default] Should always be written as zero.
0 RXS	RX Endpoint Stall - Read/Write 0 End Point OK. [Default] 1 End Point Stalled This bit is set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It is cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint,

Table continues on the next page...

USBC_*n*_ENDPTCTRL5 field descriptions (continued)

Field	Description
	Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It continues to returning STALL until this bit is either cleared by software or automatically cleared as above.

65.6.45 Endpoint Control 6 (USBC_*n*_ENDPTCTRL6)

This is endpoint control register for endpoint 6 in device operation mode.

NOTE

If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (that is Bulk-type). leaving an unconfigured endpoint control causes undefined behavior for the data pid tracking on the active endpoint/direction.

Address: 218_4000h base + 1D8h offset + (512d × i), where i=0d to 0d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								TXE	TXR	TXI	Reserved	TXT		TXD	TXS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								RXE	RXR	RXI	Reserved	RXT		RXD	RXS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USBC_*n*_ENDPTCTRL6 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved

Table continues on the next page...

USBC_*n*_ENDPTCTRL6 field descriptions (continued)

Field	Description
23 TXE	TX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
22 TXR	TX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device.
21 TXI	TX Data Toggle Inhibit 0 PID Sequencing Enabled. [Default] 1 PID Sequencing Disabled. This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.
20 -	This field is reserved. Reserved
19–18 TXT	TX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt
17 TXD	TX Endpoint Data Source - Read/Write 0 Dual Port Memory Buffer/DMA Engine [DEFAULT] Should always be written as 0.
16 TXS	TX Endpoint Stall - Read/Write 0 End Point OK 1 End Point Stalled This bit is set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It is 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 continues to returning STALL until this bit is either cleared by software or automatically cleared as above. **NOTE:** For CONTROL type endpoint, there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. Take care that the STALL bit is not set immediately after writing a '1' to it. Please follow this procedure: continually write this STALL bit until it is set or until a new setup has ben received by checking the associated ENDPTSETUPSTAT bit.
15–8 -	This field is reserved. Reserved
7 RXE	RX Endpoint Enable 0 Disabled [Default]

Table continues on the next page...

USBC_*n*_ENDPTCTRL6 field descriptions (continued)

Field	Description
	1 Enabled An Endpoint should be enabled only after it has been configured.
6 RXR	RX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device.
5 RXI	RX Data Toggle Inhibit 0 Disabled [Default] 1 Enabled This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.
4 -	This field is reserved. Reserved.
3–2 RXT	RX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Reserved
1 RXD	RX Endpoint Data Sink - Read/Write - TBD 0 Dual Port Memory Buffer/DMA Engine [Default] Should always be written as zero.
0 RXS	RX Endpoint Stall - Read/Write 0 End Point OK. [Default] 1 End Point Stalled This bit is set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It is 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 continues to returning STALL until this bit is either cleared by software or automatically cleared as above.

65.6.46 Endpoint Control 7 (USBC_*n*_ENDPTCTRL7)

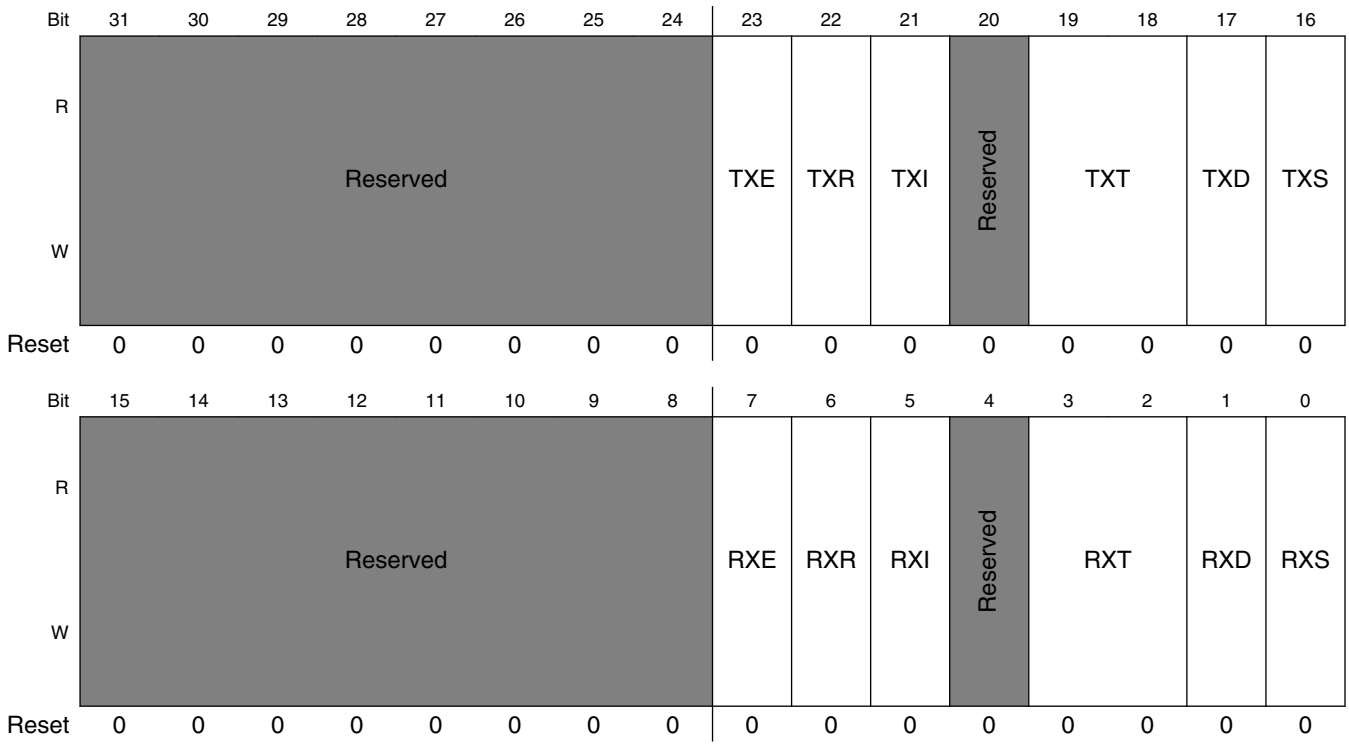
This is endpoint control register for endpoint 7 in device operation mode.

NOTE

If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (that is Bulk-type). leaving an unconfigured endpoint control

causes undefined behavior for the data pid tracking on the active endpoint/direction.

Address: 218_4000h base + 1DCh offset + (512d × i), where i=0d to 0d



USBC_*n*_ENDPTCTRL7 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23 TXE	TX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
22 TXR	TX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device.
21 TXI	TX Data Toggle Inhibit 0 PID Sequencing Enabled. [Default] 1 PID Sequencing Disabled. This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.
20 -	This field is reserved. Reserved

Table continues on the next page...

USBC_*n*_ENDPTCTRL7 field descriptions (continued)

Field	Description
19–18 TXT	TX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt
17 TXD	TX Endpoint Data Source - Read/Write 0 Dual Port Memory Buffer/DMA Engine [DEFAULT] Should always be written as 0.
16 TXS	TX Endpoint Stall - Read/Write 0 End Point OK 1 End Point Stalled This bit is set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It is 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 continues to returning STALL until this bit is either cleared by software or automatically cleared as above. **NOTE:** For CONTROL type endpoint, there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. Take care that the STALL bit is not set immediately after writing a '1' to it. Please follow this procedure: continually write this STALL bit until it is set or until a new setup has ben received by checking the associated ENDPTSETUPSTAT bit.
15–8 -	This field is reserved. Reserved
7 RXE	RX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
6 RXR	RX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device.
5 RXI	RX Data Toggle Inhibit 0 Disabled [Default] 1 Enabled This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.
4 -	This field is reserved. Reserved.
3–2 RXT	RX Endpoint Type - Read/Write 00 Control

Table continues on the next page...

USBC_*n*_ENDPTCTRL7 field descriptions (continued)

Field	Description
	01 Isochronous 10 Bulk 11 Reserved
1 RXD	RX Endpoint Data Sink - Read/Write - TBD 0 Dual Port Memory Buffer/DMA Engine [Default] Should always be written as zero.
0 RXS	RX Endpoint Stall - Read/Write 0 End Point OK. [Default] 1 End Point Stalled This bit is set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It is 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 continues to returning STALL until this bit is either cleared by software or automatically cleared as above.

Chapter 66
Universal Serial Bus 2.0 Integrated PHY (USB-PHY)

66.1 USB PHY Overview

The chip contains 2 integrated USB 2.0 PHY macrocells capable of connecting to USB host/device systems at the USB low-speed (LS) rate of 1.5 Mbits/s, full-speed (FS) rate of 12 Mbits/s or at the USB 2.0 high-speed (HS) rate of 480 Mbits/s.

See Figure 66-1 for a block diagram of the PHY. The integrated PHY provides a standard UTM interface. The USB_*n*_DN and USB_*n*_DP pins connect directly to a USB connector.

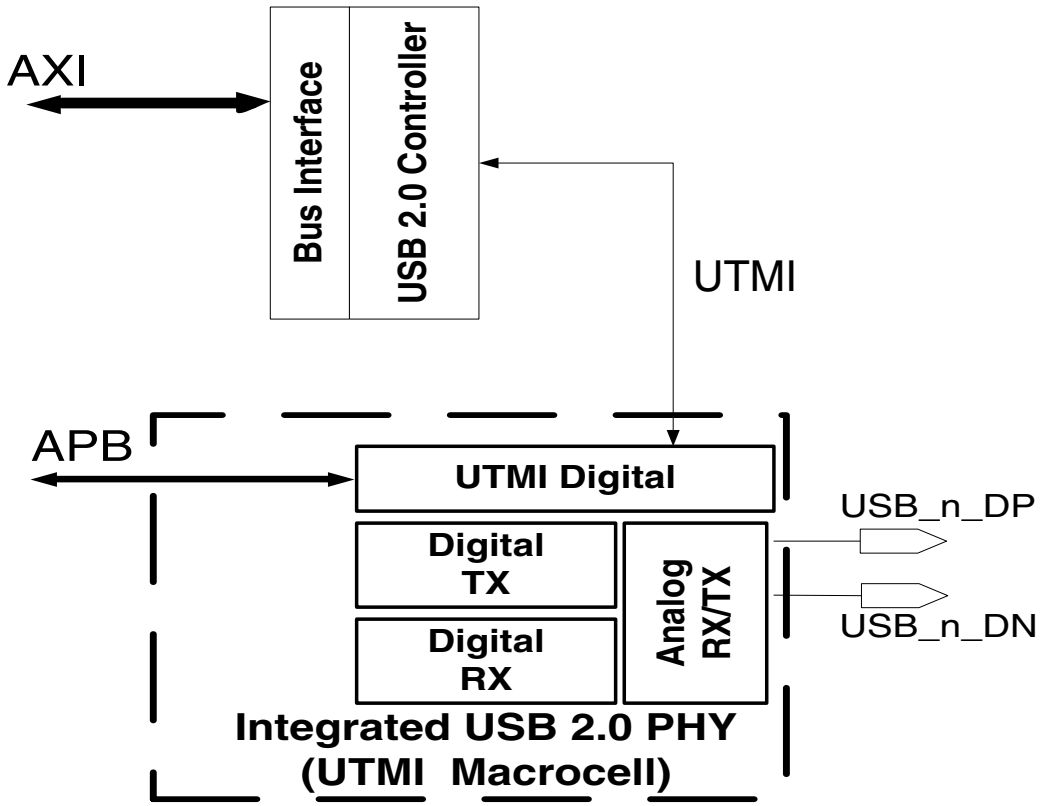


Figure 66-1. USB 2.0 PHY Block Diagram

USBPHY1 is the PHY interface for USB OTG controller; USBPHY2 is the PHY interface for USB Host1 controller.

The following subsections describe the external interfaces, internal interfaces, major blocks, and programmable registers that comprise the integrated USB 2.0 PHY.

66.2 Operation

The UTM provides a 16-bit interface to the USB controller. This interface is clocked at 30 MHz.

- The digital portions of the USBPHY block include the UTMI, digital transmitter, digital receiver, and the programmable registers.
- The analog transceiver section comprises an analog receiver and an analog transmitter, as shown in Figure 66-2.

66.2.1 UTMI

The UTMI block handles the line_state bits, reset buffering, suspend distribution, transceiver speed selection, and transceiver termination selection.

The PLL supplies a 120 MHz signal to all of the digital logic. The UTMI block does a final divide-by-four to develop the 30 MHz clock used in the interface.

66.2.2 Digital Transmitter

The digital transmitter receives the 16-bit transmit data from the USB controller and handles the tx_valid, tx_validh and tx_ready handshake.

In addition, it contains the transmit serializer that converts the 16-bit parallel words at 30 MHz to a single bitstream at 480 Mbit for high-speed or 12 Mbit for full-speed or 1.5 Mbit for low-speed. It does this while implementing the bit-stuffing algorithm and the NRZI encoder that are used to remove the DC component from the serial bitstream. The output of this encoder is sent to the low-speed (LS), full-speed (FS) or high-speed (HS) drivers in the analog transceiver section's transmitter block.

66.2.3 Digital Receiver

The digital receiver receives the raw serial bitstream from the low speed (LS) differential transceiver, full speed (FS) differential transceiver, and a 9X, 480 MHz sampled data from the high speed (HS) differential transceiver.

As the phase of the USB host transmitter shifts relative to the local PLL, the receiver section's HS DLL tracks these changes to give a reliable sample of the incoming 480 Mbit/s bitstream. Since this sample point shifts relative to the PLL phase used by the digital logic, a rate-matching elastic buffer is provided to cross this clock domain boundary. Once the bitstream is in the local clock domain, an NRZI decoder and bit unstuffer restore the original payload data bitstream and pass it to a deserializer and holding register. The receive state machine handles the rx_valid, rx_validh, and handshake with the USB controller. The handshake is not interlocked, in that there is no rx_ready signal coming from the controller. The controller must take each 16-bit value as presented by the PHY. The receive state machine provides an rx_active signal to the controller that indicates when it is inside a valid packet (SYNC detected, and so on).

66.2.4 Analog Receiver

The analog receiver comprises five differential receivers, two single-ended receivers, and a 9X, 480 MHz HS data sampling module

, as shown in the figure below and described further in this section.

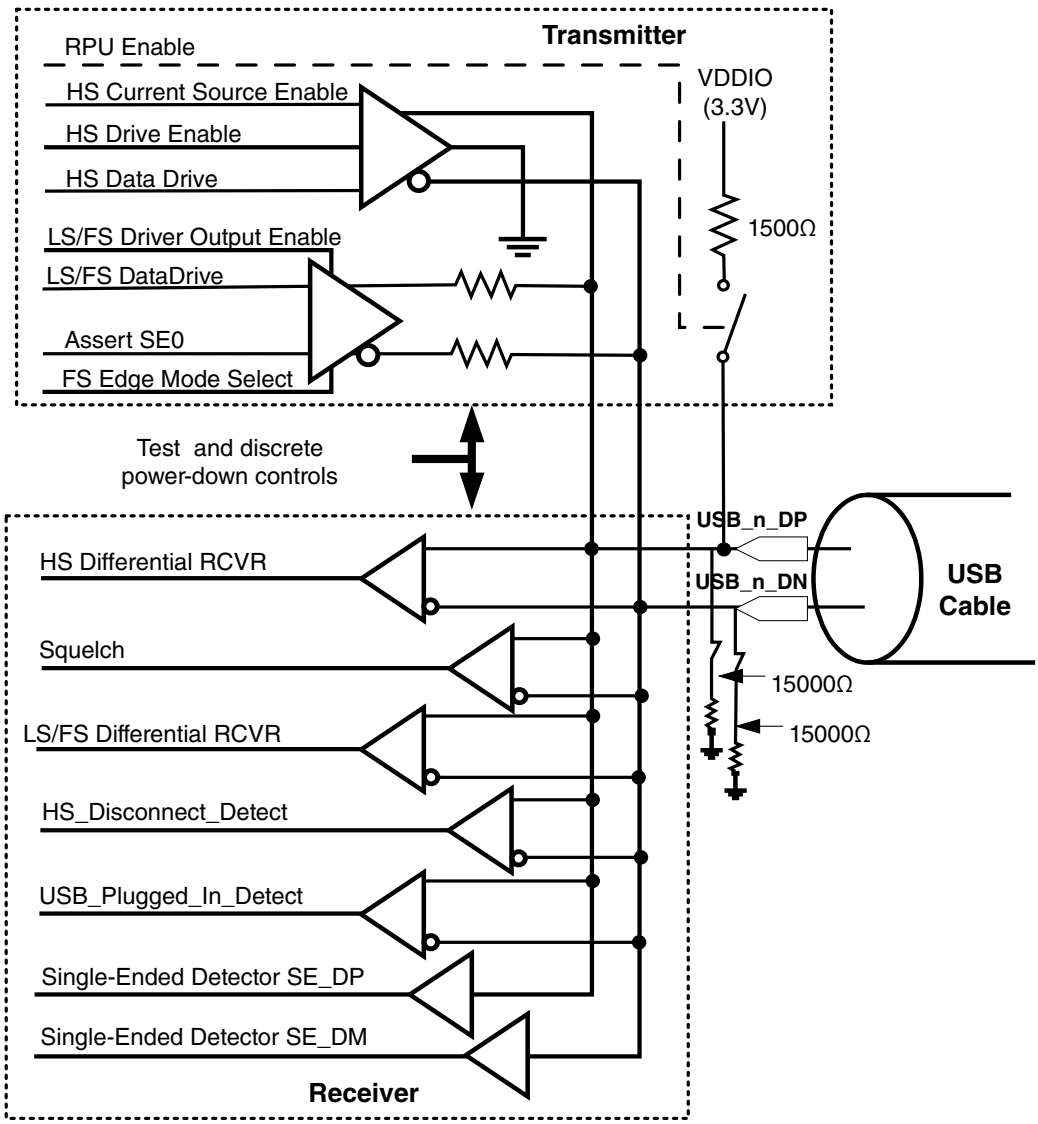


Figure 66-2. USB 2.0 PHY Analog Transceiver Block Diagram

66.2.4.1 HS Differential Receiver

The high-speed differential receiver is both a differential analog receiver and threshold comparator. Its output is a one if the differential signal is greater than a 0-V threshold.

Otherwise, its output is 0. Its purpose is to discriminate the ± 400-mV differential voltage resulting from the high-speed drivers current flow into the dual 45Ω terminations found on each pin of the differential pair. The envelope or squelch detector, described below, ensures that the differential signal has sufficient magnitude to be valid. The HS differential receiver tolerates up to 500 mV of common mode offset.

66.2.4.2 Squelch Detector

The squelch detector is a differential analog receiver and threshold comparator.

Its output is 1, if the differential magnitude is less than a nominal 100 mV threshold. Otherwise, its output is 0.

Its purpose is to invalidate the HS differential receiver when the incoming signal is simply too low to receive reliably.

66.2.4.3 LS/FS Differential Receiver

The low-speed/full-speed differential receiver is both a differential analog receiver and threshold comparator.

The crossover voltage falls between 1.3 V and 2.0 V. Its output is 1, when the USB_*n*_DP line is above the crossover point and the USB_*n*_DN line is below the crossover point. The digital receiver section decodes the receiver data into J or K state according to the speed.

66.2.4.4 HS Disconnect Detector

It is a differential analog receiver and threshold comparator. It outputs high when differential magnitude is greater than a nominal 575-mV threshold. Otherwise, it outputs low.

66.2.4.5 USB Plugged-In Detector

The USB plugged-in detector looks for both USB_*n*_DP and USB_*n*_DN to be high. There is a pair of large on-chip pullup resistors (200 KΩ) that hold both USB_*n*_DP and USB_*n*_DN high when the USB cable is not attached. The USB plugged-in detector signals a 0 in this case.

When operating in device mode, the upstream port in host/hub interface contains a 15 KΩ pulldown resistor which could easily override the 200 KΩ pullup resistor. When plugged in, at least one signal in the pair will be low, which will force the plugged-in detector's output high.

66.2.4.6 Single-Ended USB_DP Receiver

The single-ended USB_*n*_DP receiver output is high whenever the USB_*n*_DP input is above its nominal 1.8 V threshold.

66.2.4.7 Single-Ended USB_DN Receiver

The single-ended USB_*n*_DN receiver output is high whenever the USB_*n*_DN input is above its nominal 1.8 V threshold.

66.2.4.8 9X Oversample Module

The 9X oversample module uses nine identically spaced phases of the 480 MHz clock to sample a high speed bit data. The squelch signal is sampled only 1X.

66.2.5 Analog Transmitter

The analog transmitter comprises two differential drivers: one for high-speed signaling and one for full-speed signaling. It also contains the switchable 1.5 KΩ pullup resistor.

See Figure 66-2.

66.2.5.1 Switchable High-Speed 45Ω Termination Resistors

High-speed current mode differential signaling requires good 90 Ω differential termination at each end of the USB cable. This results from switching in 45 Ω terminating resistors from each signal line to ground at each end of the cable.

Because each signal is parallel terminated with 45 Ω at each end, each driver sees a 22.5 Ω load. This load impedance is much too low for full-speed signaling levels—hence the need for switchable high-speed terminating resistors. Switchable trimming resistors are provided to tune the actual termination resistance of each device, as shown in Figure

66-3. The HW_USBPHY_TX_TXCAL45DP bit field, for example, allows one of 16 trimming resistor values to be placed in parallel with the 45Ω terminator on the USB_*n*_DP signal.

66.2.5.2 Low-Speed/Full-Speed Differential Driver

The low-speed/full-speed differential drivers are essentially low-impedance pulldown devices that are switched in a differential mode for low-speed or full-speed signaling, that is, either one or the other device is turned on to signal the "J" state or the "K" state.

66.2.5.3 High-Speed Differential Driver

The high-speed differential driver receives a 17.78 mA current from the constant current source (Iref) and essentially steers it down either the USB_DP signal or the USB_DN signal or alternatively to ground.

This current will produce approximately a 400 mV drop across the 22.5 Ω termination seen by the driver when it is steered onto one of the signal lines. The approximately 17.78 mA current source is referenced back to the integrated voltage-band-gap (Vbg) circuit. The Iref, Ibias, and V to I circuits are shared with the integrated battery charger.

66.2.5.4 Switchable 1.5KΩ USB_DP Pullup Resistor

This product contains a switchable 1.5 KΩ pullup resistor on the USB_*n*_DP signal.

This resistor is switched on to indicate to the host/hub controller that a full-speed-capable device is on the USB cable, powered on, and ready. This resistor is switched off at power-on reset so the host does not recognize a USB device until the processor software enables the announcement of a full-speed device.

66.2.5.5 Switchable 15KΩ USB_DP Pulldown Resistor

This product contains a switchable 15 KΩ pulldown resistor on both USB_*n*_DP and USB_*n*_DN signals. This is used in host mode to indicate to the device controller that a host is present.

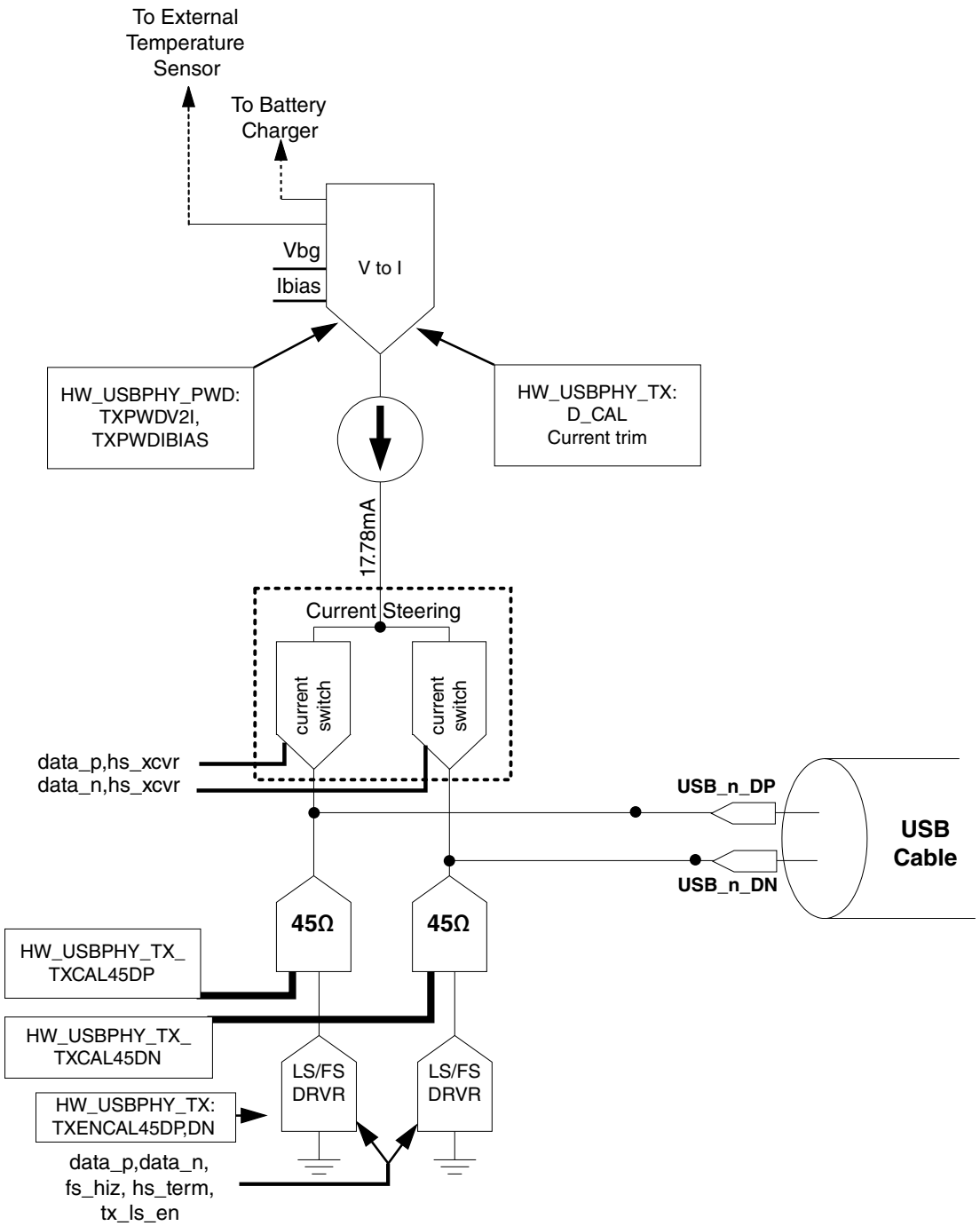


Figure 66-3. USB 2.0 PHY Transmitter Block Diagram

66.2.6 Recommended Register Configuration for USB Certification

The register settings in this section are recommended for passing USB certification.

The following settings lower the J/K levels to certifiable limits:

HW_USBPHY_TX_TXCAL45DP = 0x0
HW_USBPHY_TX_TXCAL45DN = 0x0
HW_USBPHY_TX_D_CAL = 0x7

66.2.7 Charger detection

The USB charger detector is a block that detects whether the upstream-facing device is connected to a down-stream facing charger, either a dedicated USB charger or a host charger.

The USB charger detector is comprised of two sub-blocks, namely the USB data-pin contact detector and the charger detector.

This section details those two sub-blocks and gives the software flow of USB charger detection. Finally, this chapter discusses the detection of a USB charger in case of a dead battery.

66.2.7.1 Charger detect control table

Before we dive into the details of the detectors, we show the logic table of the control signals to give the user an overall picture of the charger detector.

Table 66-1. Charger detection control table

EN_B	CHK_CHRG_B	CHK_CONTACT	Data pin contact detector	Charger detector
0	1	1	Enabled	Disabled
0	0	x(don't care)	Disabled	Enabled
1	x	x(don't care)	Disabled	Disabled

66.2.7.2 Data pin contact detector

According to Battery Charging Specification (rev 1.2), USB plugs and receptacles are designed such that when the plug is inserted into the receptacle, the power pins make contact before the data pins make contact. Therefore, there is inevitably a time interval during which USB_*n*_VBUS has been observed by the device while the USB_*n*_DP and USB_*n*_DN pins are not still pending for contact. The USB data pin contact detector is designed to give the software an indication of the contact of the data pins.

To enable the USB data pin contact detector, the user should set the CHK_CONTACT bit of the USB1_CHRG_DETECT register to 1 and monitor the PLUG_CONTACT bit status of the USB1_CHRG_DETECT_STAT register. If PLUG_CONTACT is 1, then it indicates that the data pins have make good contacts, otherwise the user should continue to wait until this bit is set.

According to Table 1, it should be noted that the data pin contact detector only works when EN_B=0 and CHK_CHRG_B=1, both bit being of the USB1_CHRG_DETECT register.

66.2.7.3 Charger detector

Once the data pins make contact, the user should enable the charger detector by clearing the CHK_CHRG_B bit that is low-active. Then the user should wait for 40ms and then check the status bit of CHRG_DETECTED in register hw_anadig_usb1_chrg_det_stat. CHRG_DETECTED=1 means that the device is connected to a charger, either a dedicated charger or a charging downstream port (or equivalently called a host charger, or charging host). To further differentiate between a host charger and a dedicated charger, the user is suggested to pull up USB_*n*_DP signaling a connect event to the host. Then the user should monitor the USB_*n*_DN line status. If USB_*n*_DN=1, then the charger is a dedicated charger; if USB_*n*_DN=0, then it is a host charger.

66.2.7.4 Charger detection software flow

Upon seeing VBUS, the software should follow the software flow for the charger detection process. The flow chart mentions the "enable the vdd3p0 current limiter". Please refer to the power chapter for details.

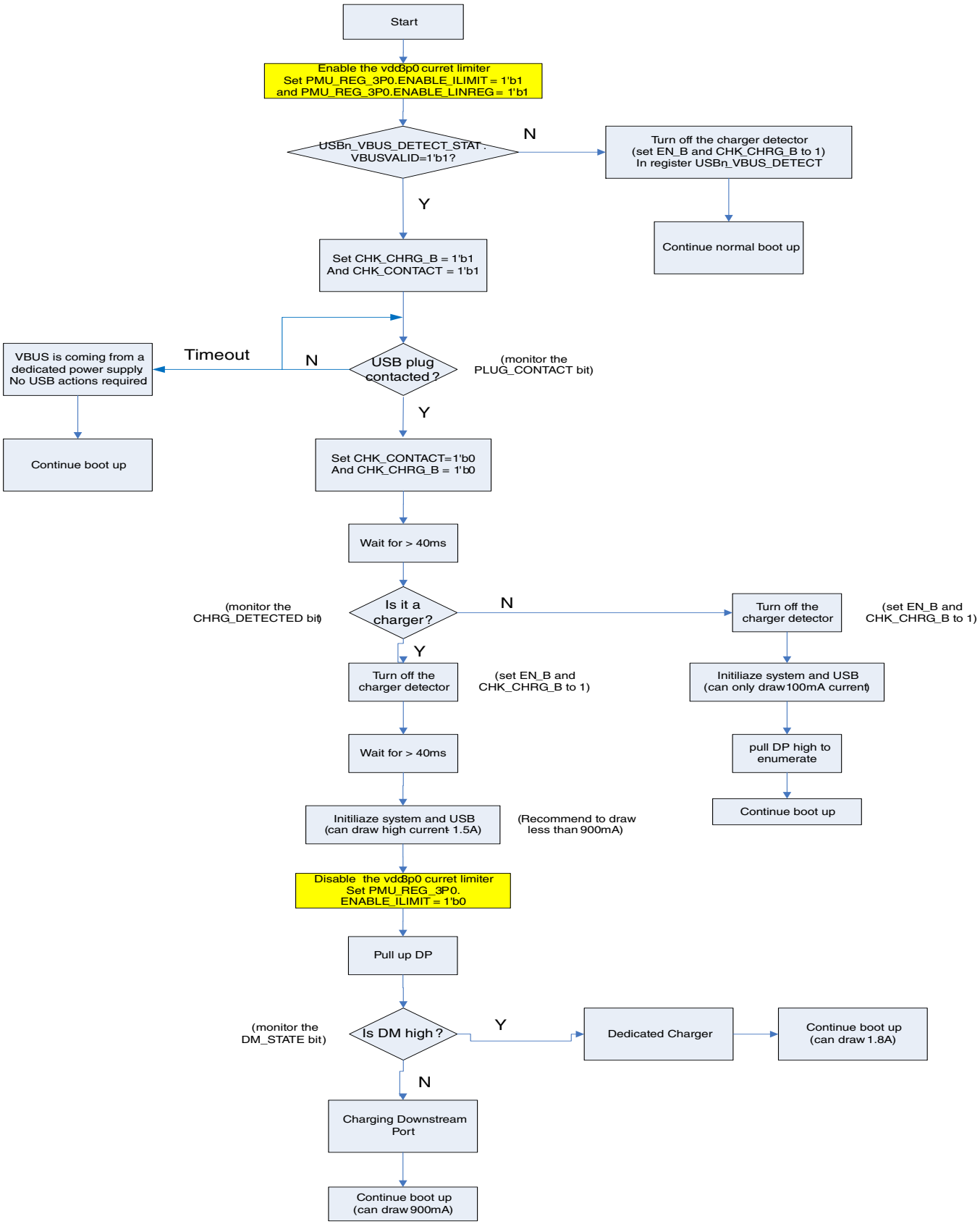


Figure 66-4. USBPHY Charger Detection Software Flow

66.2.7.5 Dead Battery Protect

All the descriptions above are based on the assumption that all the power supplies have been on when the device is plugged into a remote host (or charger). However, there are cases when the local battery of the portable device has been so depleted that the system could not be turned on. In such scenarios the user may prefer a method of signaling the external power management unit (PMIC) the existence of the USB charger to draw a current larger than 100mA from the remote host to speed up system boot up or battery charging. The charger detector indeed supports this function.

When we have a fully depleted battery, all the power supplies might be off. Upon insertion of the 5V, the supplies are brought up by the external PMIC and the internal regulators. Due to the 100mA inrush current limit of the USB spec, we cannot draw larger than 100mA current which might be a limit for system boot-up. Since by default, EN_B=0, CHK_CHRG_B=0 and CHK_CONTACT=1, the usb charger detector is automatically enabled without any software operation needed and it can signal the external PMIC the existence of a USB charger through the open-drain output pin USB_OTG_CHD_B. This pin should be pulled up to an external voltage that is acceptable to the PMIC. If this signal is low, then the PMIC can get that the device is connected to a charger. In this case, the PMIC can draw more than 100mA current from the USB.

It should be noted that this function requires cooperation between the chip and the external PMIC. It is suggested that the user consult Freescale for such use cases.

66.3 USB PHY Memory Map/Register Definition

USBPHY Hardware Register Format Summary

USBPHY memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20C_9000	USB PHY Power-Down Register (USBPHY1_PWD)	32	R/W	001E_1C00h	66.3.1/5529
20C_9004	USB PHY Power-Down Register (USBPHY1_PWD_SET)	32	R/W	001E_1C00h	66.3.1/5529
20C_9008	USB PHY Power-Down Register (USBPHY1_PWD_CLR)	32	R/W	001E_1C00h	66.3.1/5529
20C_900C	USB PHY Power-Down Register (USBPHY1_PWD_TOG)	32	R/W	001E_1C00h	66.3.1/5529
20C_9010	USB PHY Transmitter Control Register (USBPHY1_TX)	32	R/W	1006_0607h	66.3.2/5531

Table continues on the next page...

USBPHY memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20C_9014	USB PHY Transmitter Control Register (USBPHY1_TX_SET)	32	R/W	1006_0607h	66.3.2/5531
20C_9018	USB PHY Transmitter Control Register (USBPHY1_TX_CLR)	32	R/W	1006_0607h	66.3.2/5531
20C_901C	USB PHY Transmitter Control Register (USBPHY1_TX_TOG)	32	R/W	1006_0607h	66.3.2/5531
20C_9020	USB PHY Receiver Control Register (USBPHY1_RX)	32	R/W	0000_0000h	66.3.3/5532
20C_9024	USB PHY Receiver Control Register (USBPHY1_RX_SET)	32	R/W	0000_0000h	66.3.3/5532
20C_9028	USB PHY Receiver Control Register (USBPHY1_RX_CLR)	32	R/W	0000_0000h	66.3.3/5532
20C_902C	USB PHY Receiver Control Register (USBPHY1_RX_TOG)	32	R/W	0000_0000h	66.3.3/5532
20C_9030	USB PHY General Control Register (USBPHY1_CTRL)	32	R/W	C020_0000h	66.3.4/5534
20C_9034	USB PHY General Control Register (USBPHY1_CTRL_SET)	32	R/W	C020_0000h	66.3.4/5534
20C_9038	USB PHY General Control Register (USBPHY1_CTRL_CLR)	32	R/W	C020_0000h	66.3.4/5534
20C_903C	USB PHY General Control Register (USBPHY1_CTRL_TOG)	32	R/W	C020_0000h	66.3.4/5534
20C_9040	USB PHY Status Register (USBPHY1_STATUS)	32	R/W	0000_0000h	66.3.5/5537
20C_9050	USB PHY Debug Register (USBPHY1_DEBUG)	32	R/W	7F18_0000h	66.3.6/5539
20C_9054	USB PHY Debug Register (USBPHY1_DEBUG_SET)	32	R/W	7F18_0000h	66.3.6/5539
20C_9058	USB PHY Debug Register (USBPHY1_DEBUG_CLR)	32	R/W	7F18_0000h	66.3.6/5539
20C_905C	USB PHY Debug Register (USBPHY1_DEBUG_TOG)	32	R/W	7F18_0000h	66.3.6/5539
20C_9060	UTMI Debug Status Register 0 (USBPHY1_DEBUG0_STATUS)	32	R	0000_0000h	66.3.7/5541
20C_9070	UTMI Debug Status Register 1 (USBPHY1_DEBUG1)	32	R/W	0000_1000h	66.3.8/5542
20C_9074	UTMI Debug Status Register 1 (USBPHY1_DEBUG1_SET)	32	R/W	0000_1000h	66.3.8/5542
20C_9078	UTMI Debug Status Register 1 (USBPHY1_DEBUG1_CLR)	32	R/W	0000_1000h	66.3.8/5542
20C_907C	UTMI Debug Status Register 1 (USBPHY1_DEBUG1_TOG)	32	R/W	0000_1000h	66.3.8/5542
20C_9080	UTMI RTL Version (USBPHY1_VERSION)	32	R	0402_0000h	66.3.9/5543
20C_A000	USB PHY Power-Down Register (USBPHY2_PWD)	32	R/W	001E_1C00h	66.3.1/5529
20C_A004	USB PHY Power-Down Register (USBPHY2_PWD_SET)	32	R/W	001E_1C00h	66.3.1/5529
20C_A008	USB PHY Power-Down Register (USBPHY2_PWD_CLR)	32	R/W	001E_1C00h	66.3.1/5529
20C_A00C	USB PHY Power-Down Register (USBPHY2_PWD_TOG)	32	R/W	001E_1C00h	66.3.1/5529
20C_A010	USB PHY Transmitter Control Register (USBPHY2_TX)	32	R/W	1006_0607h	66.3.2/5531
20C_A014	USB PHY Transmitter Control Register (USBPHY2_TX_SET)	32	R/W	1006_0607h	66.3.2/5531
20C_A018	USB PHY Transmitter Control Register (USBPHY2_TX_CLR)	32	R/W	1006_0607h	66.3.2/5531
20C_A01C	USB PHY Transmitter Control Register (USBPHY2_TX_TOG)	32	R/W	1006_0607h	66.3.2/5531
20C_A020	USB PHY Receiver Control Register (USBPHY2_RX)	32	R/W	0000_0000h	66.3.3/5532

Table continues on the next page...

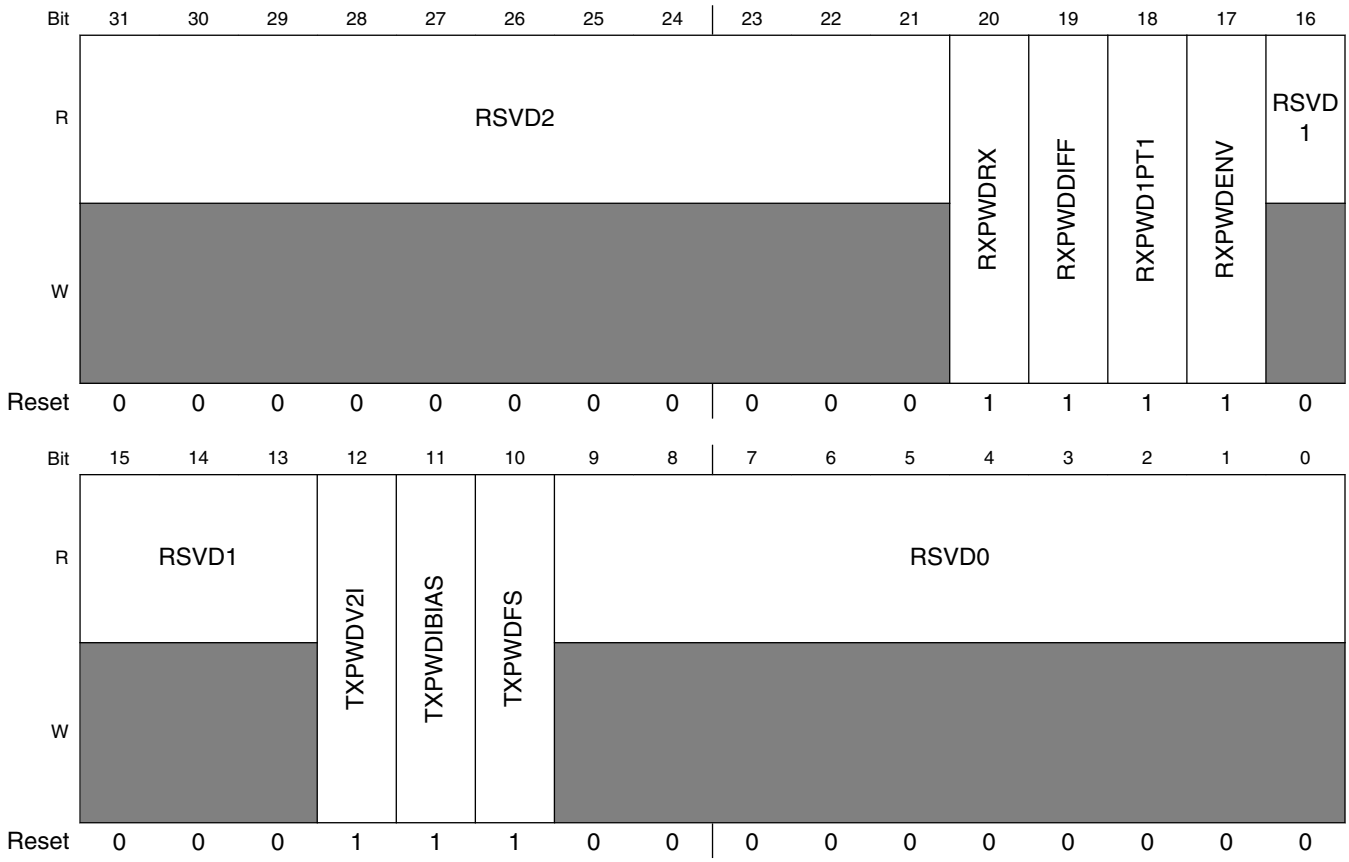
USBPHY memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20C_A024	USB PHY Receiver Control Register (USBPHY2_RX_SET)	32	R/W	0000_0000h	66.3.3/5532
20C_A028	USB PHY Receiver Control Register (USBPHY2_RX_CLR)	32	R/W	0000_0000h	66.3.3/5532
20C_A02C	USB PHY Receiver Control Register (USBPHY2_RX_TOG)	32	R/W	0000_0000h	66.3.3/5532
20C_A030	USB PHY General Control Register (USBPHY2_CTRL)	32	R/W	C020_0000h	66.3.4/5534
20C_A034	USB PHY General Control Register (USBPHY2_CTRL_SET)	32	R/W	C020_0000h	66.3.4/5534
20C_A038	USB PHY General Control Register (USBPHY2_CTRL_CLR)	32	R/W	C020_0000h	66.3.4/5534
20C_A03C	USB PHY General Control Register (USBPHY2_CTRL_TOG)	32	R/W	C020_0000h	66.3.4/5534
20C_A040	USB PHY Status Register (USBPHY2_STATUS)	32	R/W	0000_0000h	66.3.5/5537
20C_A050	USB PHY Debug Register (USBPHY2_DEBUG)	32	R/W	7F18_0000h	66.3.6/5539
20C_A054	USB PHY Debug Register (USBPHY2_DEBUG_SET)	32	R/W	7F18_0000h	66.3.6/5539
20C_A058	USB PHY Debug Register (USBPHY2_DEBUG_CLR)	32	R/W	7F18_0000h	66.3.6/5539
20C_A05C	USB PHY Debug Register (USBPHY2_DEBUG_TOG)	32	R/W	7F18_0000h	66.3.6/5539
20C_A060	UTMI Debug Status Register 0 (USBPHY2_DEBUG0_STATUS)	32	R	0000_0000h	66.3.7/5541
20C_A070	UTMI Debug Status Register 1 (USBPHY2_DEBUG1)	32	R/W	0000_1000h	66.3.8/5542
20C_A074	UTMI Debug Status Register 1 (USBPHY2_DEBUG1_SET)	32	R/W	0000_1000h	66.3.8/5542
20C_A078	UTMI Debug Status Register 1 (USBPHY2_DEBUG1_CLR)	32	R/W	0000_1000h	66.3.8/5542
20C_A07C	UTMI Debug Status Register 1 (USBPHY2_DEBUG1_TOG)	32	R/W	0000_1000h	66.3.8/5542
20C_A080	UTMI RTL Version (USBPHY2_VERSION)	32	R	0402_0000h	66.3.9/5543

66.3.1 USB PHY Power-Down Register (USBPHYx_PWD*n*)

The USB PHY Power-Down Register provides overall control of the PHY power state. Before programming this register, the PHY clocks must be enabled in registers USBPHYx_CTRLn and CCM_ANALOG_USBPHYx_PLL_480_CTRLn.

Address: Base address + 0h offset + (4d × i), where i=0d to 3d



USBPHYx_PWD*n* field descriptions

Field	Description
31–21 RSVD2	Reserved.
20 RXPWDRX	0 = Normal operation. 1 = Power-down the entire USB PHY receiver block except for the full-speed differential receiver. Note that this bit will be auto cleared if there is USB wakeup event while ENAUTOCLR_PHY_PWD bit of USBPHYx_CTRL is enabled.
19 RXPWDDIFF	0 = Normal operation. 1 = Power-down the USB high-speed differential receiver.

Table continues on the next page...

USBPHYx_PWD*n* field descriptions (continued)

Field	Description
	Note that this bit will be auto cleared if there is USB wakeup event while ENAUTOCLR_PHY_PWD bit of USBPHYx_CTRL is enabled.
18 RXPWD1PT1	0 = Normal operation. 1 = Power-down the USB full-speed differential receiver. Note that this bit will be auto cleared if there is USB wakeup event while ENAUTOCLR_PHY_PWD bit of USBPHYx_CTRL is enabled.
17 RXPWDENV	0 = Normal operation. 1 = Power-down the USB high-speed receiver envelope detector (squelch signal). Note that this bit will be auto cleared if there is USB wakeup event while ENAUTOCLR_PHY_PWD bit of USBPHYx_CTRL is enabled.
16–13 RSVD1	Reserved.
12 TXPWDV2I	0 = Normal operation. 1 = Power-down the USB PHY transmit V-to-I converter and the current mirror. Note that this bit will be auto cleared if there is USB wakeup event while ENAUTOCLR_PHY_PWD bit of USBPHYx_CTRL is enabled. Note that these circuits are shared with the battery charge circuit. Setting this to 1 does not power-down these circuits, unless the corresponding bit in the battery charger is also set for power-down.
11 TXPWDIBIAS	0 = Normal operation. 1 = Power-down the USB PHY current bias block for the transmitter. This bit should be set only when the USB is in suspend mode. This effectively powers down the entire USB transmit path. Note that this bit will be auto cleared if there is USB wakeup event while ENAUTOCLR_PHY_PWD bit of USBPHYx_CTRL is enabled. Note that these circuits are shared with the battery charge circuit. Setting this bit to 1 does not power-down these circuits, unless the corresponding bit in the battery charger is also set for power-down.
10 TXPWDFS	0 = Normal operation. 1 = Power-down the USB full-speed drivers. This turns off the current starvation sources and puts the drivers into high-impedance output. Note that this bit will be auto cleared if there is USB wakeup event while ENAUTOCLR_PHY_PWD bit of USBPHYx_CTRL is enabled.
RSVD0	Reserved.

66.3.2 USB PHY Transmitter Control Register (USBPHY*x*_TX*n*)

The USB PHY Transmitter Control Register handles the transmit controls.

Address: Base address + 10h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD5			USBPHY_TX_ EDGECTRL			RSVD2						TXCAL45DP			
W																
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1				TXCAL45DN				RSVD0				D_CAL			
W																
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	1

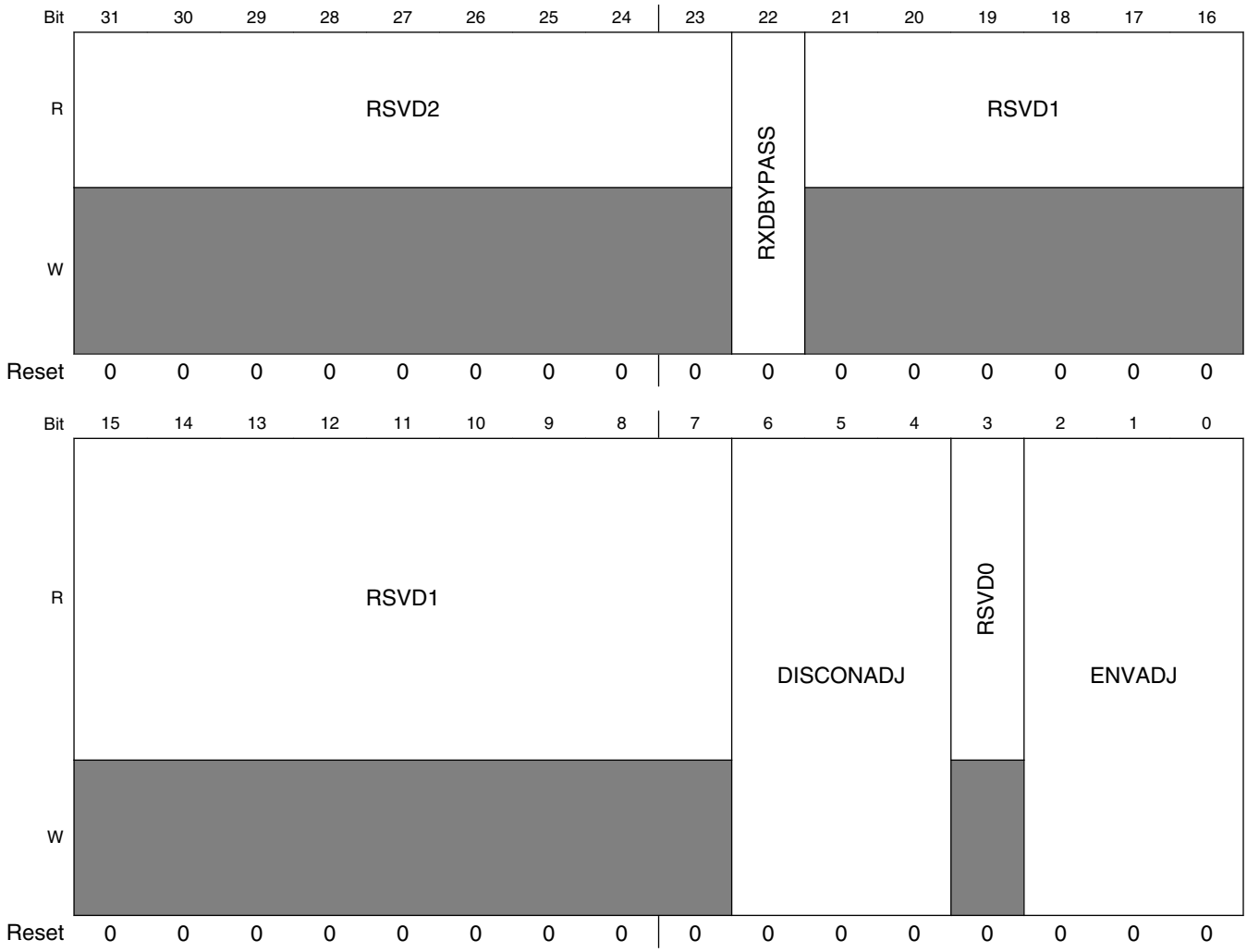
USBPHY*x*_TX*n* field descriptions

Field	Description
31–29 RSVD5	Reserved.
28–26 USBPHY_TX_ EDGECTRL	Controls the edge-rate of the current sensing transistors used in HS transmit. NOT FOR CUSTOMER USE.
25–20 RSVD2	Reserved.
19–16 TXCAL45DP	Decode to select a 45-Ohm resistance to the USB_DP output pin. Maximum resistance = 0000. Resistance is centered by design at 0110.
15–12 RSVD1	Reserved. **Note:** This bit should remain clear.
11–8 TXCAL45DN	Decode to select a 45-Ohm resistance to the USB_DN output pin. Maximum resistance = 0000. Resistance is centered by design at 0110.
7–4 RSVD0	Reserved. **Note:** This bit should remain clear.
D_CAL	Resistor Trimming Code: 0000 = 0.16% 0111 = Nominal 1111 = +25%

66.3.3 USB PHY Receiver Control Register (USBPHY*x*_RX*n*)

The USB PHY Receiver Control Register handles receive path controls.

Address: Base address + 20h offset + (4d × i), where i=0d to 3d



USBPHY*x*_RX*n* field descriptions

Field	Description
31–23 RSVD2	Reserved.
22 RXDBYPASS	0 = Normal operation. 1 = Use the output of the USB_DP single-ended receiver in place of the full-speed differential receiver. This test mode is intended for lab use only.
21–7 RSVD1	Reserved.

Table continues on the next page...

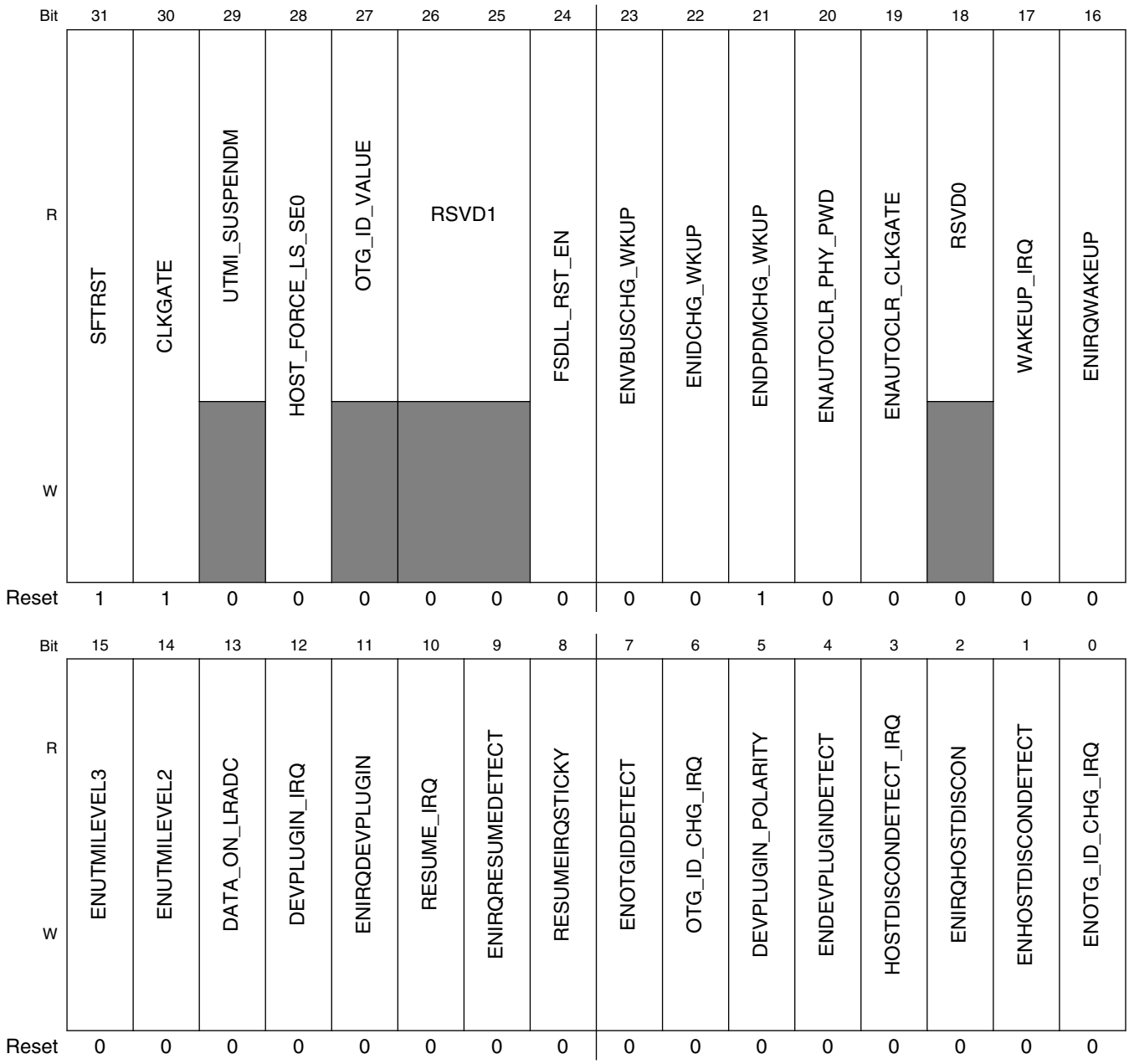
USBPHYx_RXn field descriptions (continued)

Field	Description
6–4 DISCONADJ	The DISCONADJ field adjusts the trip point for the disconnect detector: 000 = Trip-Level Voltage is 0.57500 V 001 = Trip-Level Voltage is 0.56875 V 010 = Trip-Level Voltage is 0.58125 V 011 = Trip-Level Voltage is 0.58750 V 1XX = Reserved
3 RSVD0	Reserved.
ENVADJ	The ENVADJ field adjusts the trip point for the envelope detector. 000 = Trip-Level Voltage is 0.12500 V 001 = Trip-Level Voltage is 0.10000 V 010 = Trip-Level Voltage is 0.13750 V 011 = Trip-Level Voltage is 0.15000 V 1XX = Reserved

66.3.4 USB PHY General Control Register (USBPHYx_CTRLn)

The USB PHY General Control Register handles OTG and Host controls. This register also includes interrupt enables and connectivity detect enables and results.

Address: Base address + 30h offset + (4d × i), where i=0d to 3d



USBPHYx_CTRL*n* field descriptions

Field	Description
31 SFTRST	Writing a 1 to this bit will soft-reset the USBPHYx_PWD, USBPHYx_TX, USBPHYx_RX, and USBPHYx_CTRL registers. Set to 0 to release the PHY from reset.
30 CLKGATE	Gate UTMI Clocks. Clear to 0 to run clocks. Set to 1 to gate clocks. Set this to save power while the USB is not actively being used. Configuration state is kept while the clock is gated. Note this bit can be auto-cleared if there is any wakeup event when USB is suspended while ENAUTOCLR_CLKGATE bit of USBPHYx_CTRL is enabled.
29 UTMI_SUSPENDM	Used by the PHY to indicate a powered-down state. If all the power-down bits in the USBPHYx_PWD are enabled, UTMI_SUSPENDM will be 0, otherwise 1. UTMI_SUSPENDM is negative logic, as required by the UTMI specification.
28 HOST_FORCE_LS_SE0	Forces the next FS packet that is transmitted to have a EOP with LS timing. This bit is used in host mode for the resume sequence. After the packet is transferred, this bit is cleared. The design can use this function to force the LS SE0 or use the USBPHYx_CTRL_UTMI_SUSPENDM to trigger this event when leaving suspend. This bit is used in conjunction with USBPHYx_DEBUG_HOST_RESUME_DEBUG.
27 OTG_ID_VALUE	Almost same as OTGID_STATUS in USBPHYx_STATUS Register. The only difference is that OTG_ID_VALUE has debounce logic to filter the glitches on ID Pad.
26–25 RSVD1	Reserved.
24 FSDLL_RST_EN	Enables the feature to reset the FSDLL lock detection logic at the end of each TX packet.
23 ENVBUSCHG_WKUP	Enables the feature to wakeup USB if VBUS is toggled when USB is suspended.
22 ENIDCHG_WKUP	Enables the feature to wakeup USB if ID is toggled when USB is suspended.
21 ENDPDMCHG_WKUP	Enables the feature to wakeup USB if DP/DM is toggled when USB is suspended. This bit is enabled by default.
20 ENAUTOCLR_PHY_PWD	Enables the feature to auto-clear the PWD register bits in USBPHYx_PWD if there is wakeup event while USB is suspended. This should be enabled if needs to support auto wakeup without S/W's interaction.
19 ENAUTOCLR_CLKGATE	Enables the feature to auto-clear the CLKGATE bit if there is wakeup event while USB is suspended. This should be enabled if needs to support auto wakeup without S/W's interaction.
18 RSVD0	Reserved.
17 WAKEUP_IRQ	Indicates that there is a wakeup event. Reset this bit by writing a 1 to the clear address space and not by a general write.
16 ENIRQWAKEUP	Enables interrupt for the wakeup events.
15 ENUTMILEVEL3	Enables UTMI+ Level3. This should be enabled if needs to support external FS Hub with LS device connected
14 ENUTMILEVEL2	Enables UTMI+ Level2. This should be enabled if needs to support LS device
13 DATA_ON_LRADC	Enables the LRADC to monitor USB_DP and USB_DM. This is for use in non-USB modes only.
12 DEVPLUGIN_IRQ	Indicates that the device is connected. Reset this bit by writing a 1 to the clear address space and not by a general write.
11 ENIRQDEVPLUGIN	Enables interrupt for the detection of connectivity to the USB line.

Table continues on the next page...

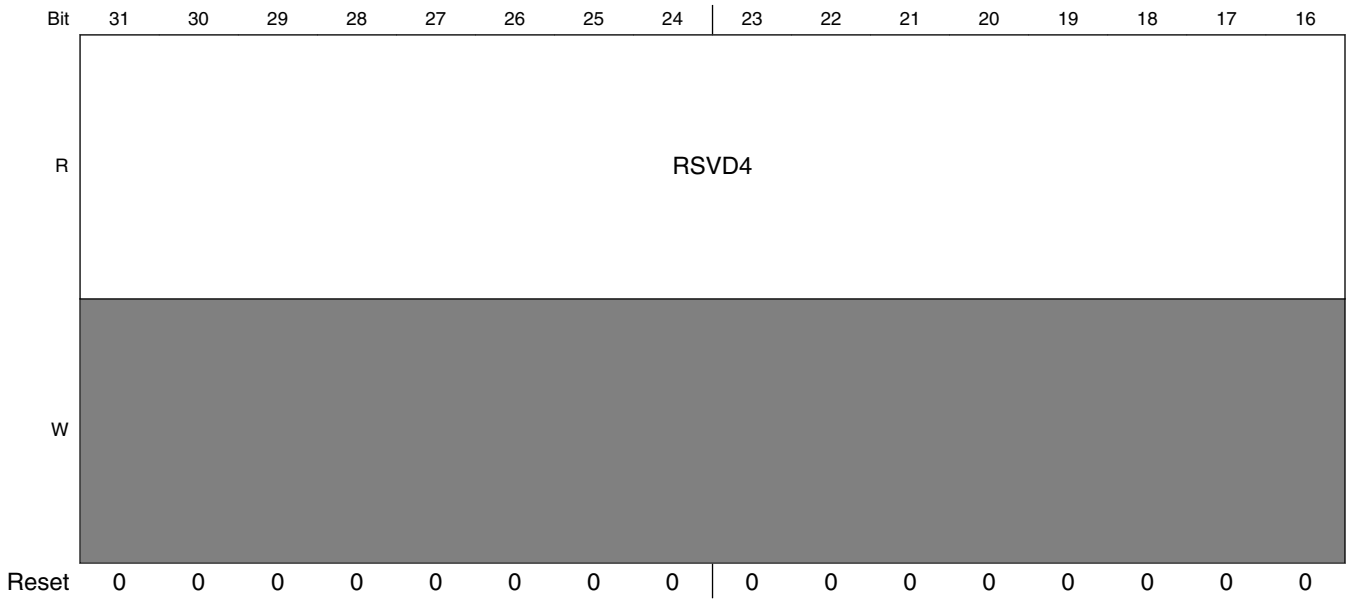
USBPHY*x*_CTRL*n* field descriptions (continued)

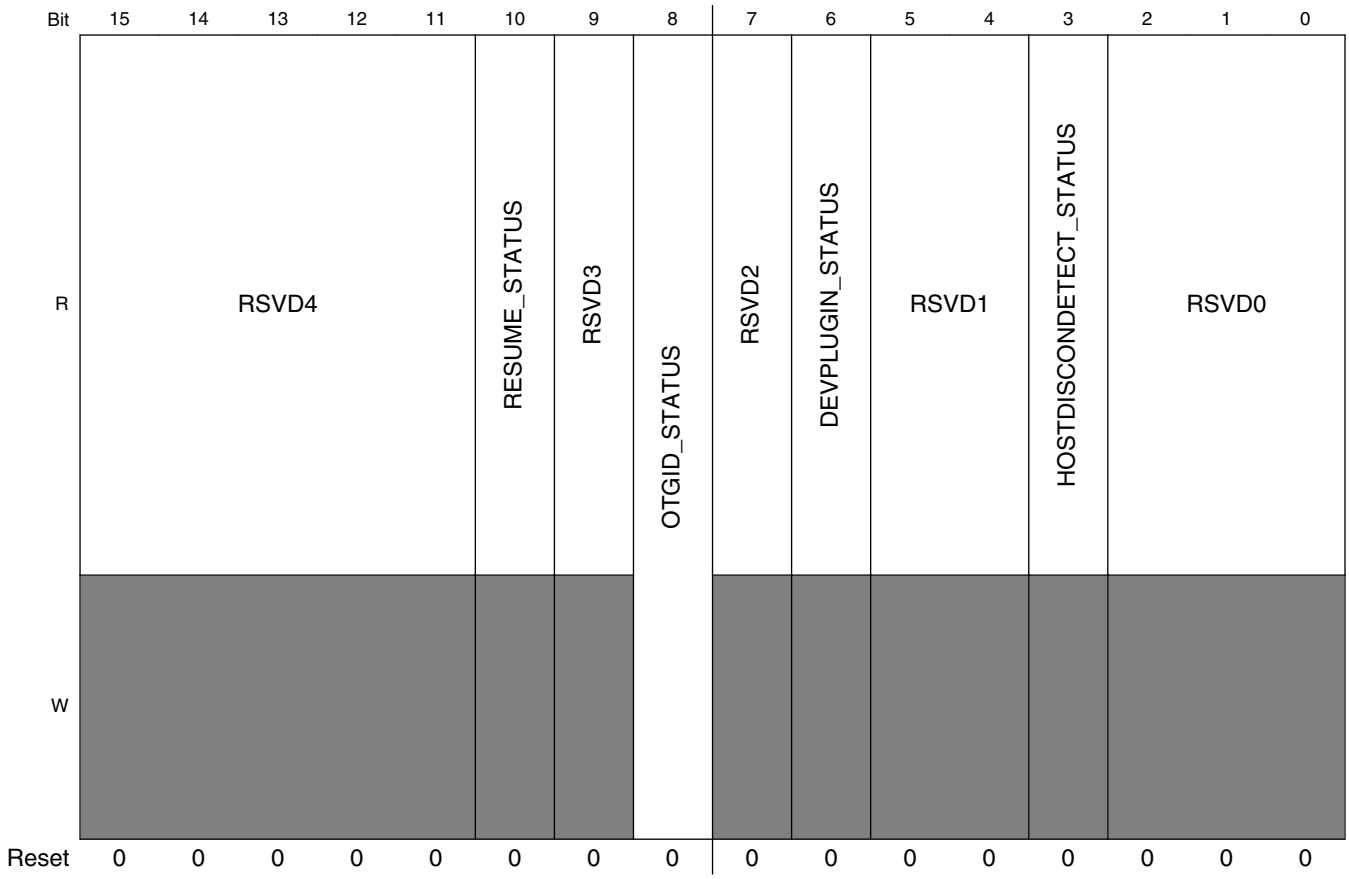
Field	Description
10 RESUME_IRQ	Indicates that the host is sending a wake-up after suspend. This bit is also set on a reset during suspend. Use this bit to wake up from suspend for either the resume or the reset case. Reset this bit by writing a 1 to the clear address space and not by a general write.
9 ENIRQRESUMEDETECT	Enables interrupt for detection of a non-J state on the USB line. This should only be enabled after the device has entered suspend mode.
8 RESUMEIRQSTICKY	Set to 1 will make RESUME_IRQ bit a sticky bit until software clear it. Set to 0, RESUME_IRQ only set during the wake-up period.
7 ENOTGIDDETECT	Enables circuit to detect resistance of MiniAB ID pin.
6 OTG_ID_CHG_IRQ	OTG ID change interrupt. Indicates the value of ID pin changed.
5 DEVPLUGIN_POLARITY	For device mode, if this bit is cleared to 0, then it trips the interrupt if the device is plugged in. If set to 1, then it trips the interrupt if the device is unplugged.
4 ENDEVPLUGINDETECT	For device mode, enables 200-KOhm pullups for detecting connectivity to the host.
3 HOSTDISCONDETECT_ IRQ	Indicates that the device has disconnected in high-speed mode. Reset this bit by writing a 1 to the clear address space and not by a general write.
2 ENIRQHOSTDISCON	Enables interrupt for detection of disconnection to Device when in high-speed host mode. This should be enabled after ENDEVPLUGINDETECT is enabled.
1 ENHOSTDISCONDETECT	For host mode, enables high-speed disconnect detector. This signal allows the override of enabling the detection that is normally done in the UTMI controller. The UTMI controller enables this circuit whenever the host sends a start-of-frame packet. SW shall set this bit when it found the high-speed device is connected, suggested during bus reset, after found high-speed device in USB_PORTSC1.PSPD). SW shall make sure this bit is not set at the end of resume, otherwise a wrong disconnect status may be detected. Suggest clear it after set USB_PORTSC1.SUSP, set it again after resume is ended(USB_PORTSC1.FPR==0).
0 ENOTG_ID_CHG_IRQ	Enable OTG_ID_CHG_IRQ.

66.3.5 USB PHY Status Register (USBPHYx_STATUS)

The USB PHY Status Register holds results of IRQ and other detects.

Address: Base address + 40h offset





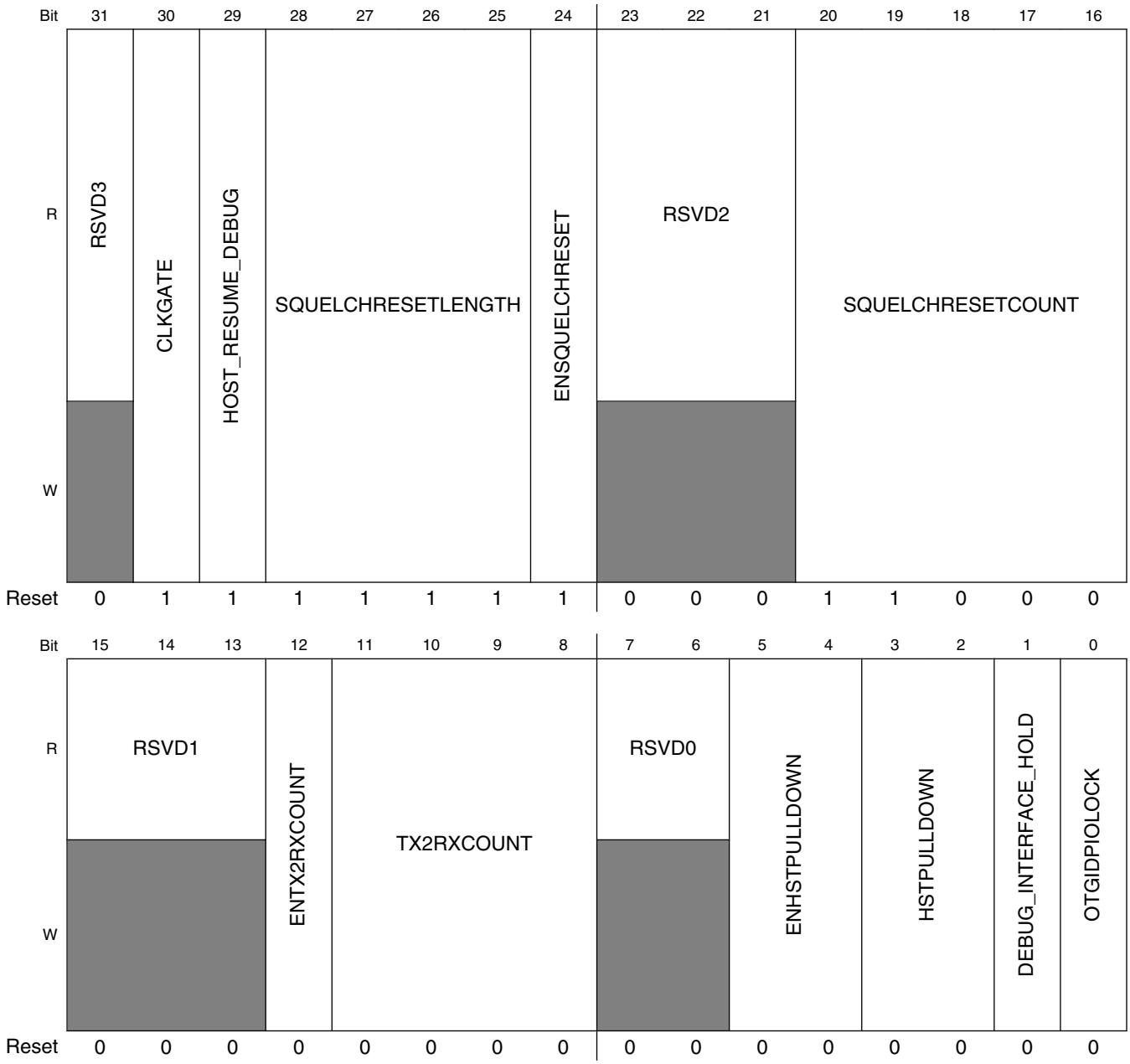
USBPHYx_STATUS field descriptions

Field	Description
31–11 RSVD4	Reserved.
10 RESUME_STATUS	Indicates that the host is sending a wake-up after suspend and has triggered an interrupt.
9 RSVD3	Reserved.
8 OTGID_STATUS	Indicates the results of ID pin on MiniAB plug. False (0) is when ID resistance is less than Ra_Plug_ID, indicating host (A) side. True (1) is when ID resistance is greater than Rb_Plug_ID, indicating device (B) side.
7 RSVD2	Reserved.
6 DEVPLUGIN_STATUS	Indicates that the device has been connected on the USB_DP and USB_DM lines.
5–4 RSVD1	Reserved.
3 HOSTDISCONDETECT_ STATUS	Indicates that the device has disconnected while in high-speed host mode.
RSVD0	Reserved.

66.3.6 USB PHY Debug Register (USBPHY*x*_DEBUG*n*)

This register is used to debug the USB PHY.

Address: Base address + 50h offset + (4d × i), where i=0d to 3d



USBPHYx_DEBUG*n* field descriptions

Field	Description
31 RSVD3	Reserved.
30 CLKGATE	Gate Test Clocks. Clear to 0 for running clocks. Set to 1 to gate clocks. Set this to save power while the USB is not actively being used. Configuration state is kept while the clock is gated.
29 HOST_RESUME_DEBUG	Choose to trigger the host resume SE0 with HOST_FORCE_LS_SE0 = 0 or UTMI_SUSPEND = 1.
28–25 SQUELCHRESETLENGTH	Duration of RESET in terms of the number of 480-MHz cycles.
24 ENSQUELCHRESET	Set bit to allow squelch to reset high-speed receive.
23–21 RSVD2	Reserved.
20–16 SQUELCHRESETCOUNT	Delay in between the detection of squelch to the reset of high-speed RX.
15–13 RSVD1	Reserved.
12 ENTX2RXCOUNT	Set this bit to allow a countdown to transition in between TX and RX.
11–8 TX2RXCOUNT	Delay in between the end of transmit to the beginning of receive. This is a Johnson count value and thus will count to 8.
7–6 RSVD0	Reserved.
5–4 ENHSTPULLDOWN	Set bit 5 to 1 to override the control of the USB_DP 15-KOhm pulldown. Set bit 4 to 1 to override the control of the USB_DM 15-KOhm pulldown. Clear to 0 to disable.
3–2 HSTPULLDOWN	Set bit 3 to 1 to pull down 15-KOhm on USB_DP line. Set bit 2 to 1 to pull down 15-KOhm on USB_DM line. Clear to 0 to disable.
1 DEBUG_INTERFACE_HOLD	Use holding registers to assist in timing for external UTMI interface.
0 OTGIDPIOLOCK	Once OTG ID from USBPHYx_STATUS_OTGID_STATUS, use this to hold the value. This is to save power for the comparators that are used to determine the ID status.

66.3.7 UTMI Debug Status Register 0 (USBPHYx_DEBUG0_STATUS)

The UTMI Debug Status Register 0 holds multiple views for counters and status of state machines. This is used in conjunction with the USBPHYx_DEBUG1_DBG_ADDRESS field to choose which function to view. The default is described in the bit fields below and is used to count errors.

Address: Base address + 60h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SQUELCH_COUNT						UTMI_RXERROR_FAIL_COUNT										LOOP_BACK_FAIL_COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USBPHYx_DEBUG0_STATUS field descriptions

Field	Description
31–26 SQUELCH_ COUNT	Running count of the squelch reset instead of normal end for HS RX.
25–16 UTMI_ RXERROR_ FAIL_COUNT	Running count of the UTMI_RXERROR.
LOOP_BACK_ FAIL_COUNT	Running count of the failed pseudo-random generator loopback. Each time entering testmode, counter goes to 900D and will count up for every detected packet failure in digital/analog loopback tests.

66.3.8 UTMI Debug Status Register 1 (USBPHYx_DEBUG1*n*)

Chooses the muxing of the debug register to be shown in USBPHYx_DEBUG0_STATUS.

Address: Base address + 70h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1																ENTAILADJVD			RSVD0											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

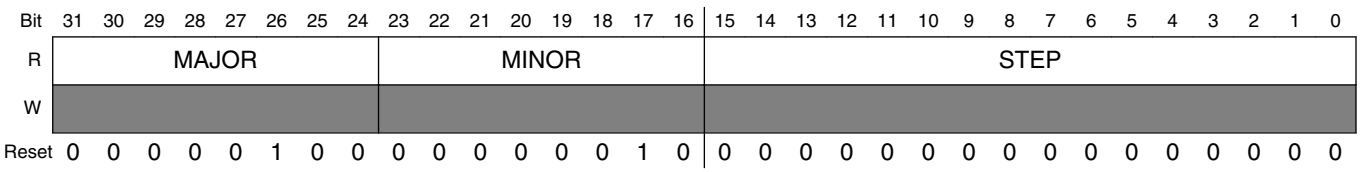
USBPHYx_DEBUG1*n* field descriptions

Field	Description
31–15 RSVD1	Reserved.
14–13 ENTAILADJVD	Delay increment of the rise of squelch: 00 = Delay is nominal 01 = Delay is +20% 10 = Delay is -20% 11 = Delay is -40%
RSVD0	Reserved. **Note:** This bit should remain clear.

66.3.9 UTMI RTL Version (USBPHYx_VERSION)

Fields for RTL Version.

Address: Base address + 80h offset



USBPHYx_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.
STEP	Fixed read-only value reflecting the stepping of the RTL version.

66.4 USB Analog Memory Map/Register Definition

USB_ANALOG memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20C_81A0	USB VBUS Detect Register (USB_ANALOG_USB1_VBUS_DETECT)	32	R/W	0010_0004h	66.4.1/5545
20C_81A4	USB VBUS Detect Register (USB_ANALOG_USB1_VBUS_DETECT_SET)	32	R/W	0010_0004h	66.4.1/5545
20C_81A8	USB VBUS Detect Register (USB_ANALOG_USB1_VBUS_DETECT_CLR)	32	R/W	0010_0004h	66.4.1/5545
20C_81AC	USB VBUS Detect Register (USB_ANALOG_USB1_VBUS_DETECT_TOG)	32	R/W	0010_0004h	66.4.1/5545
20C_81B0	USB Charger Detect Register (USB_ANALOG_USB1_CHRG_DETECT)	32	R/W	0000_0000h	66.4.2/5546
20C_81B4	USB Charger Detect Register (USB_ANALOG_USB1_CHRG_DETECT_SET)	32	R/W	0000_0000h	66.4.2/5546
20C_81B8	USB Charger Detect Register (USB_ANALOG_USB1_CHRG_DETECT_CLR)	32	R/W	0000_0000h	66.4.2/5546
20C_81BC	USB Charger Detect Register (USB_ANALOG_USB1_CHRG_DETECT_TOG)	32	R/W	0000_0000h	66.4.2/5546

Table continues on the next page...

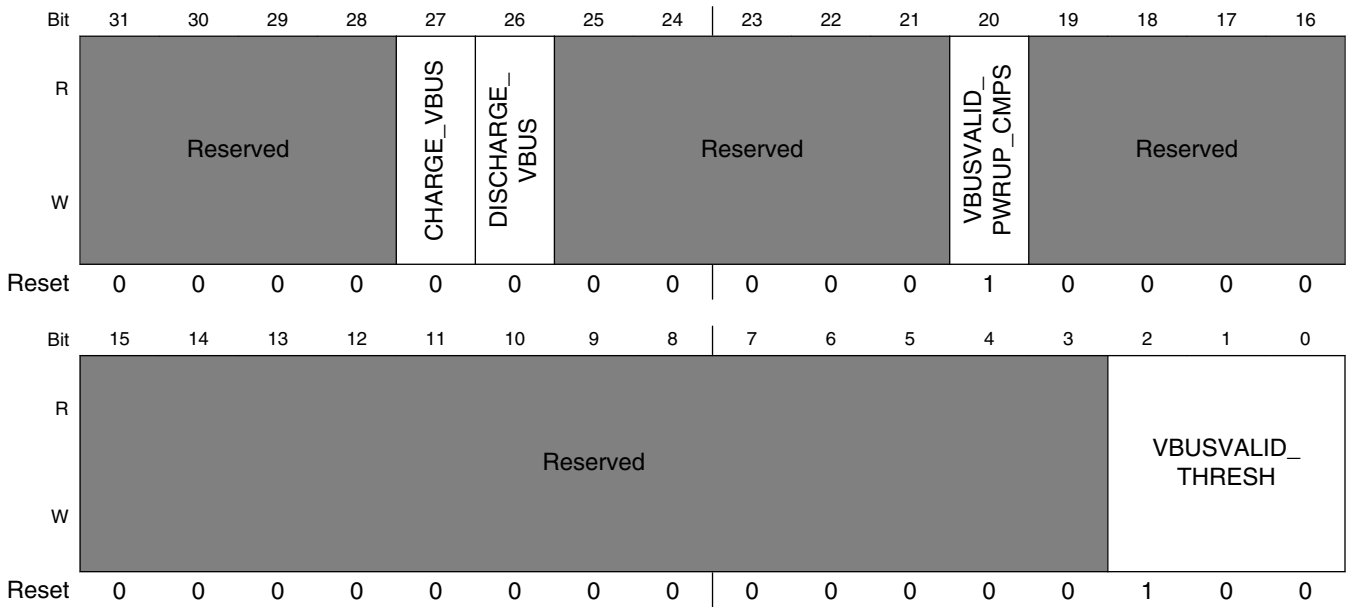
USB_ANALOG memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20C_81C0	USB VBUS Detect Status Register (USB_ANALOG_USB1_VBUS_DETECT_STAT)	32	R	0000_0000h	66.4.3/5548
20C_81D0	USB Charger Detect Status Register (USB_ANALOG_USB1_CHRG_DETECT_STAT)	32	R	0000_0000h	66.4.4/5550
20C_81F0	USB Misc Register (USB_ANALOG_USB1_MISC)	32	R/W	0000_0002h	66.4.5/5551
20C_81F4	USB Misc Register (USB_ANALOG_USB1_MISC_SET)	32	R/W	0000_0002h	66.4.5/5551
20C_81F8	USB Misc Register (USB_ANALOG_USB1_MISC_CLR)	32	R/W	0000_0002h	66.4.5/5551
20C_81FC	USB Misc Register (USB_ANALOG_USB1_MISC_TOG)	32	R/W	0000_0002h	66.4.5/5551
20C_8200	USB VBUS Detect Register (USB_ANALOG_USB2_VBUS_DETECT)	32	R/W	0010_0004h	66.4.6/5552
20C_8204	USB VBUS Detect Register (USB_ANALOG_USB2_VBUS_DETECT_SET)	32	R/W	0010_0004h	66.4.6/5552
20C_8208	USB VBUS Detect Register (USB_ANALOG_USB2_VBUS_DETECT_CLR)	32	R/W	0010_0004h	66.4.6/5552
20C_820C	USB VBUS Detect Register (USB_ANALOG_USB2_VBUS_DETECT_TOG)	32	R/W	0010_0004h	66.4.6/5552
20C_8210	USB Charger Detect Register (USB_ANALOG_USB2_CHRG_DETECT)	32	R/W	0000_0000h	66.4.7/5554
20C_8214	USB Charger Detect Register (USB_ANALOG_USB2_CHRG_DETECT_SET)	32	R/W	0000_0000h	66.4.7/5554
20C_8218	USB Charger Detect Register (USB_ANALOG_USB2_CHRG_DETECT_CLR)	32	R/W	0000_0000h	66.4.7/5554
20C_821C	USB Charger Detect Register (USB_ANALOG_USB2_CHRG_DETECT_TOG)	32	R/W	0000_0000h	66.4.7/5554
20C_8220	USB VBUS Detect Status Register (USB_ANALOG_USB2_VBUS_DETECT_STAT)	32	R	0000_0000h	66.4.8/5556
20C_8230	USB Charger Detect Status Register (USB_ANALOG_USB2_CHRG_DETECT_STAT)	32	R	0000_0000h	66.4.9/5558
20C_8250	USB Misc Register (USB_ANALOG_USB2_MISC)	32	R/W	0000_0002h	66.4.10/ 5559
20C_8254	USB Misc Register (USB_ANALOG_USB2_MISC_SET)	32	R/W	0000_0002h	66.4.10/ 5559
20C_8258	USB Misc Register (USB_ANALOG_USB2_MISC_CLR)	32	R/W	0000_0002h	66.4.10/ 5559
20C_825C	USB Misc Register (USB_ANALOG_USB2_MISC_TOG)	32	R/W	0000_0002h	66.4.10/ 5559
20C_8260	Chip Silicon Version (USB_ANALOG_DIGPROG)	32	R	0000_0000h	66.4.11/ 5560

66.4.1 USB VBUS Detect Register (USB_ANALOG_USB1_VBUS_DETECT*n*)

This register defines controls for USB VBUS detect.

Address: 20C_8000h base + 1A0h offset + (4d × i), where i=0d to 3d



USB_ANALOG_USB1_VBUS_DETECT*n* field descriptions

Field	Description
31–28 -	This field is reserved. Reserved.
27 CHARGE_VBUS	USB OTG charge VBUS.
26 DISCHARGE_ VBUS	USB OTG discharge VBUS.
25–21 -	This field is reserved. Reserved.
20 VBUSVALID_ PWRUP_CMPS	Powers up comparators for vbus_valid detector.
19–3 -	This field is reserved. Reserved.
VBUSVALID_ THRESH	Set the threshold for the VBUSVALID comparator. This comparator is the most accurate method to determine the presence of 5v, and includes hystersis to minimize the need for software debounce of the detection. This comparator has ~50mV of hystersis to prevent chattering at the comparator trip point. 000 **4V0** — 4.0V

Table continues on the next page...

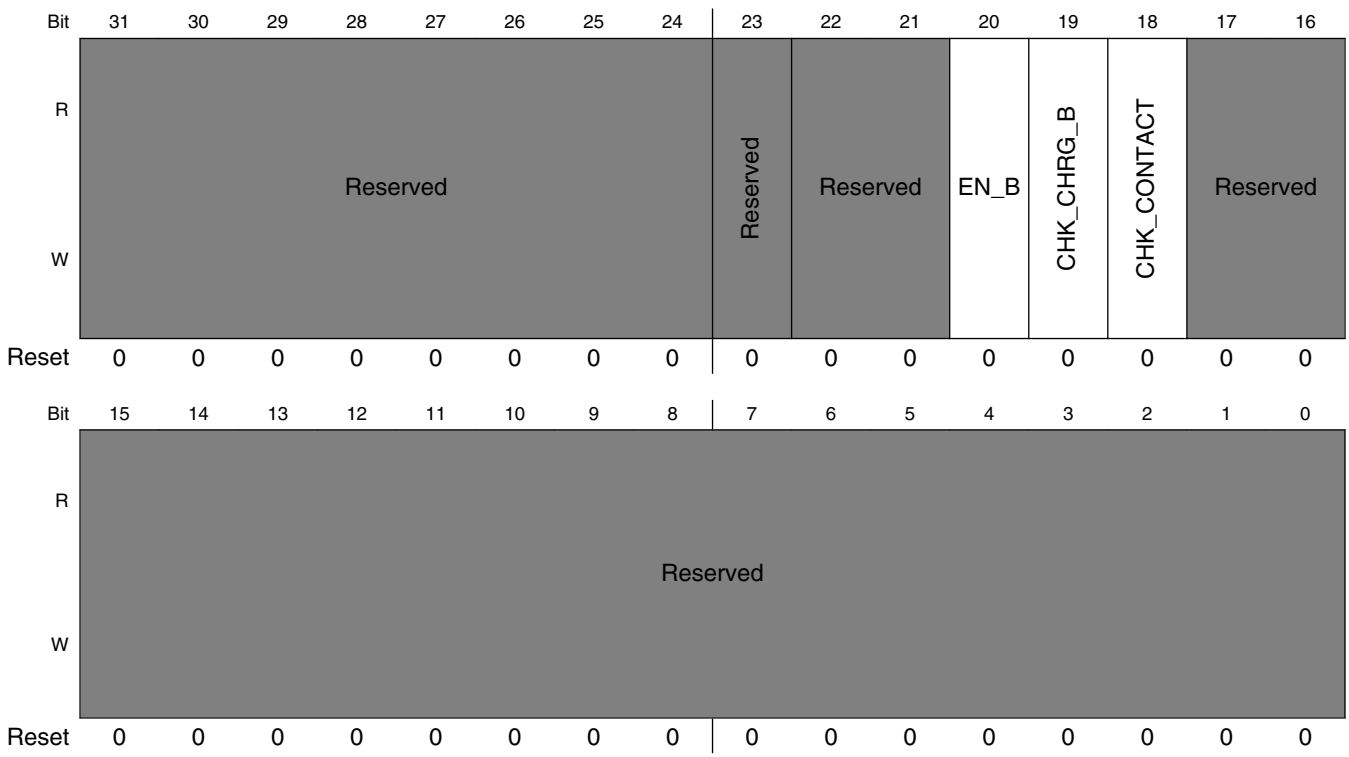
USB_ANALOG_USB1_VBUS_DETECT*n* field descriptions (continued)

Field	Description
	001 **4V1** — 4.1V 010 **4V2** — 4.2V 011 **4V3** — 4.3V 100 **4V4** — 4.4V (default) 101 **4V5** — 4.5V 110 **4V6** — 4.6V 111 **4V7** — 4.7V

66.4.2 USB Charger Detect Register (USB_ANALOG_USB1_CHRG_DETECT*n*)

This register defines controls for USB charger detect.

Address: 20C_8000h base + 1B0h offset + (4d × i), where i=0d to 3d



USB_ANALOG_USB1_CHRG_DETECT*n* field descriptions

Field	Description
31–24 -	This field is reserved. Reserved.

Table continues on the next page...

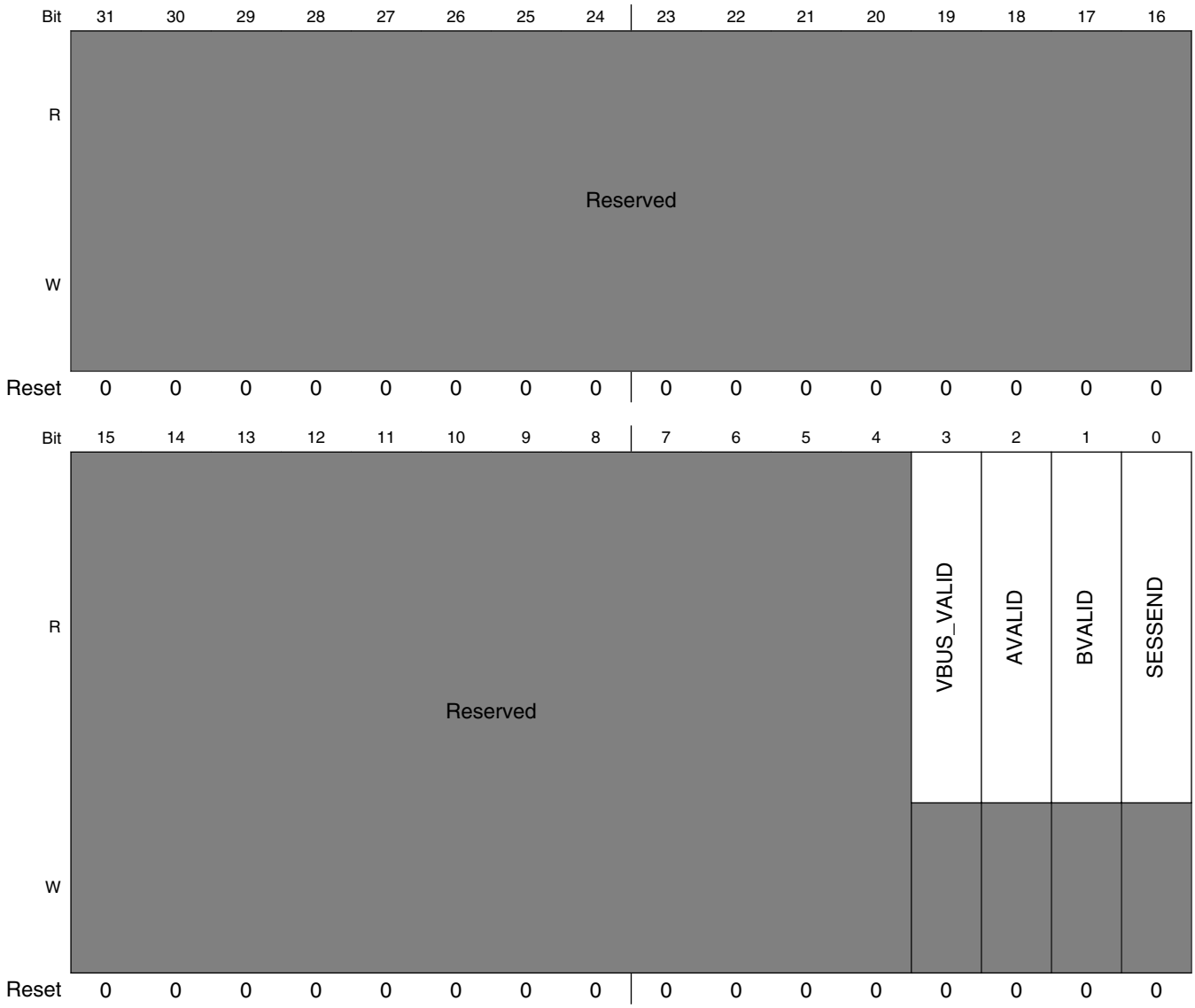
USB_ANALOG_USB1_CHRG_DETECT*n* field descriptions (continued)

Field	Description
23 -	This field is reserved. Reserved.
22–21 -	This field is reserved. Reserved.
20 EN_B	Control the charger detector. 0 **ENABLE** — Enable the charger detector. 1 **DISABLE** — Disable the charger detector.
19 CHK_CHRG_B	0 **CHECK** — Check whether a charger (either a dedicated charger or a host charger) is connected to USB port. 1 **NO_CHECK** — Do not check whether a charger is connected to the USB port.
18 CHK_CONTACT	0 **NO_CHECK** — Do not check the contact of USB plug. 1 **CHECK** — Check whether the USB plug has been in contact with each other
-	This field is reserved. Reserved.

66.4.3 USB VBUS Detect Status Register (USB_ANALOG_USB1_VBUS_DETECT_STAT)

This register defines fields for USB VBUS Detect status.

Address: 20C_8000h base + 1C0h offset = 20C_81C0h



USB_ANALOG_USB1_VBUS_DETECT_STAT field descriptions

Field	Description
31–4 -	This field is reserved. Reserved.

Table continues on the next page...

USB_ANALOG_USB1_VBUS_DETECT_STAT field descriptions (continued)

Field	Description
3 VBUS_VALID	VBus valid for USB OTG. This bit is a read only version of the state of the analog signal. It can not be overwritten by software.
2 AVALID	Indicates VBus is valid for a A-peripheral. This bit is a read only version of the state of the analog signal. It can not be overritten by software.
1 BVALID	Indicates VBus is valid for a B-peripheral. This bit is a read only version of the state of the analog signal. It can not be overritten by software.
0 SESSEND	Session End for USB OTG. This bit is a read only version of the state of the analog signal. It can not be overwritten by software like the SESSEND bit below. NOTE: This bit's default value depends on whether VDD5V is present, 0 if VDD5V is present, 1 if VDD5V is not present.

66.4.4 USB Charger Detect Status Register (USB_ANALOG_USB1_CHRG_DETECT_STAT)

This register defines fields for USB charger detect status.

Address: 20C_8000h base + 1D0h offset = 20C_81D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved												DP_STATE	DM_STATE	CHRG_DETECTED	PLUG_CONTACT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USB_ANALOG_USB1_CHRG_DETECT_STAT field descriptions

Field	Description
31–4 -	This field is reserved. Reserved.
3 DP_STATE	DP line state output of the charger detector.
2 DM_STATE	DM line state output of the charger detector.
1 CHRG_ DETECTED	State of charger detection. This bit is a read only version of the state of the analog signal. 0 **CHARGER_NOT_PRESENT** — The USB port is not connected to a charger. 1 **CHARGER_PRESENT** — A charger (either a dedicated charger or a host charger) is connected to the USB port.
0 PLUG_ CONTACT	State of the USB plug contact detector. 0 **NO_CONTACT** — The USB plug has not made contact. 1 **GOOD_CONTACT** — The USB plug has made good contact.

66.4.5 USB Misc Register (USB_ANALOG_USB1_MISC*n*)

This register defines controls for USB.

Address: 20C_8000h base + 1F0h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved	EN_CLK_UTMI	Reserved													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved														EN_DEGLITCH	HS_USE_EXTERNAL_R
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

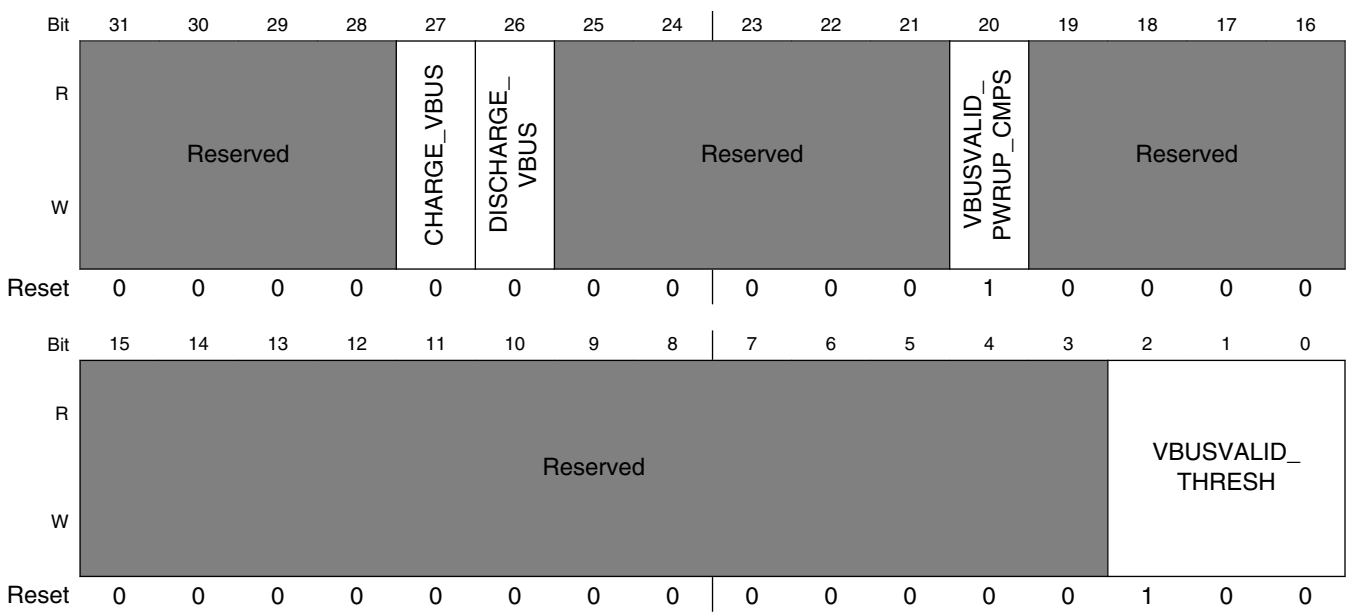
USB_ANALOG_USB1_MISC*n* field descriptions

Field	Description
31 -	This field is reserved. Reserved.
30 EN_CLK_UTMI	Enables the clk to the UTMI block.
29–2 -	This field is reserved. Reserved.
1 EN_DEGLITCH	Enable the deglitching circuit of the USB PLL output.
0 HS_USE_ EXTERNAL_R	Use external resistor to generate the current bias for the high speed transmitter. This bit should not be changed unless recommended by Freescale.

66.4.6 USB VBUS Detect Register (USB_ANALOG_USB2_VBUS_DETECT*n*)

This register defines controls for USB VBUS detect.

Address: 20C_8000h base + 200h offset + (4d × i), where i=0d to 3d



USB_ANALOG_USB2_VBUS_DETECT*n* field descriptions

Field	Description
31–28 -	This field is reserved. Reserved.

Table continues on the next page...

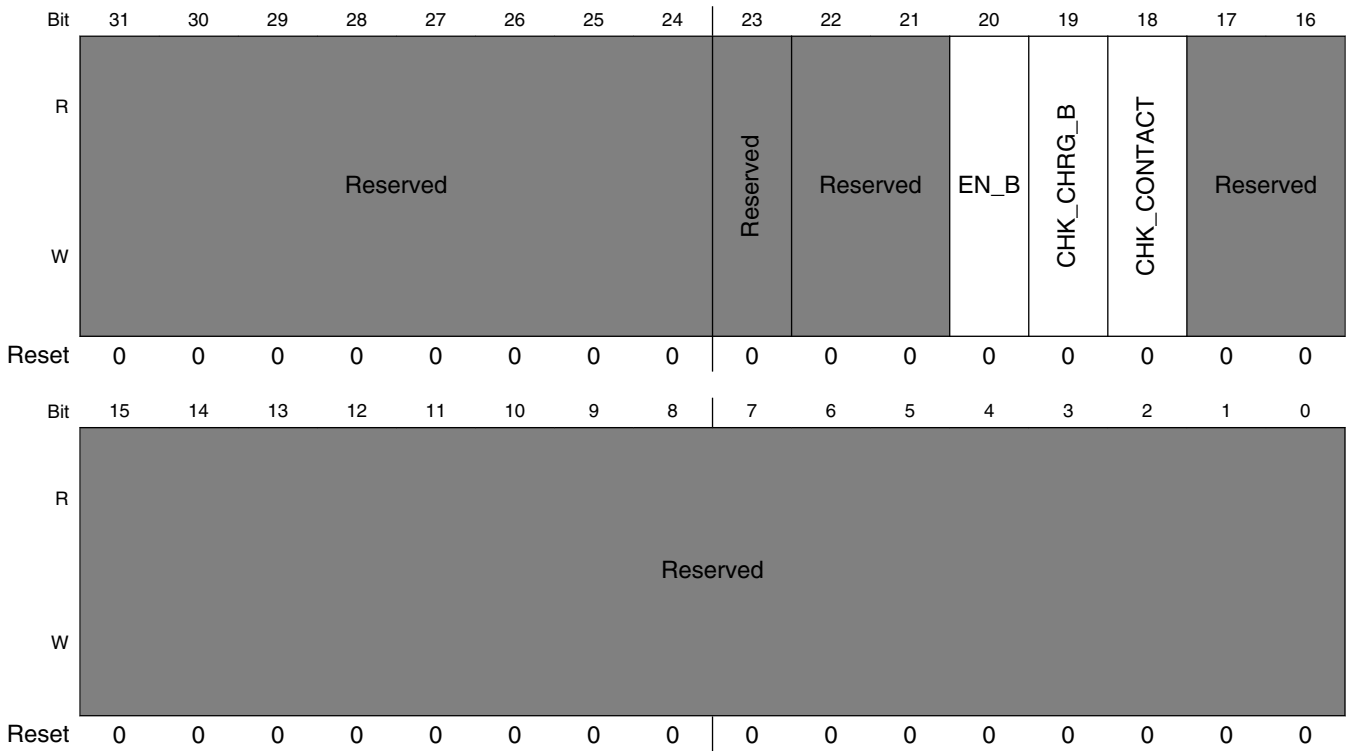
USB_ANALOG_USB2_VBUS_DETECT*n* field descriptions (continued)

Field	Description
27 CHARGE_VBUS	USB OTG charge VBUS.
26 DISCHARGE_ VBUS	USB OTG discharge VBUS.
25–21 -	This field is reserved. Reserved.
20 VBUSVALID_ PWRUP_CMPS	Powers up comparators for vbus_valid detector.
19–3 -	This field is reserved. Reserved.
VBUSVALID_ THRESH	Set the threshold for the VBUSVALID comparator. This comparator is the most accurate method to determine the presence of 5v, and includes hystersis to minimize the need for software debounce of the detection. This comparator has ~50mV of hystersis to prevent chattering at the comparator trip point. 000 **4V0** — 4.0V 001 **4V1** — 4.1V 010 **4V2** — 4.2V 011 **4V3** — 4.3V 100 **4V4** — 4.4V (default) 101 **4V5** — 4.5V 110 **4V6** — 4.6V 111 **4V7** — 4.7V

66.4.7 USB Charger Detect Register (USB_ANALOG_USB2_CHRG_DETECT*n*)

This register defines controls for USB charger detect.

Address: 20C_8000h base + 210h offset + (4d × i), where i=0d to 3d



USB_ANALOG_USB2_CHRG_DETECT*n* field descriptions

Field	Description
31–24 -	This field is reserved. Reserved.
23 -	This field is reserved. Reserved.
22–21 -	This field is reserved. Reserved.
20 EN_B	Control the charger detector. 0 **ENABLE** — Enable the charger detector. 1 **DISABLE** — Disable the charger detector.
19 CHK_CHRG_B	0 **CHECK** — Check whether a charger (either a dedicated charger or a host charger) is connected to USB port. 1 **NO_CHECK** — Do not check whether a charger is connected to the USB port.

Table continues on the next page...

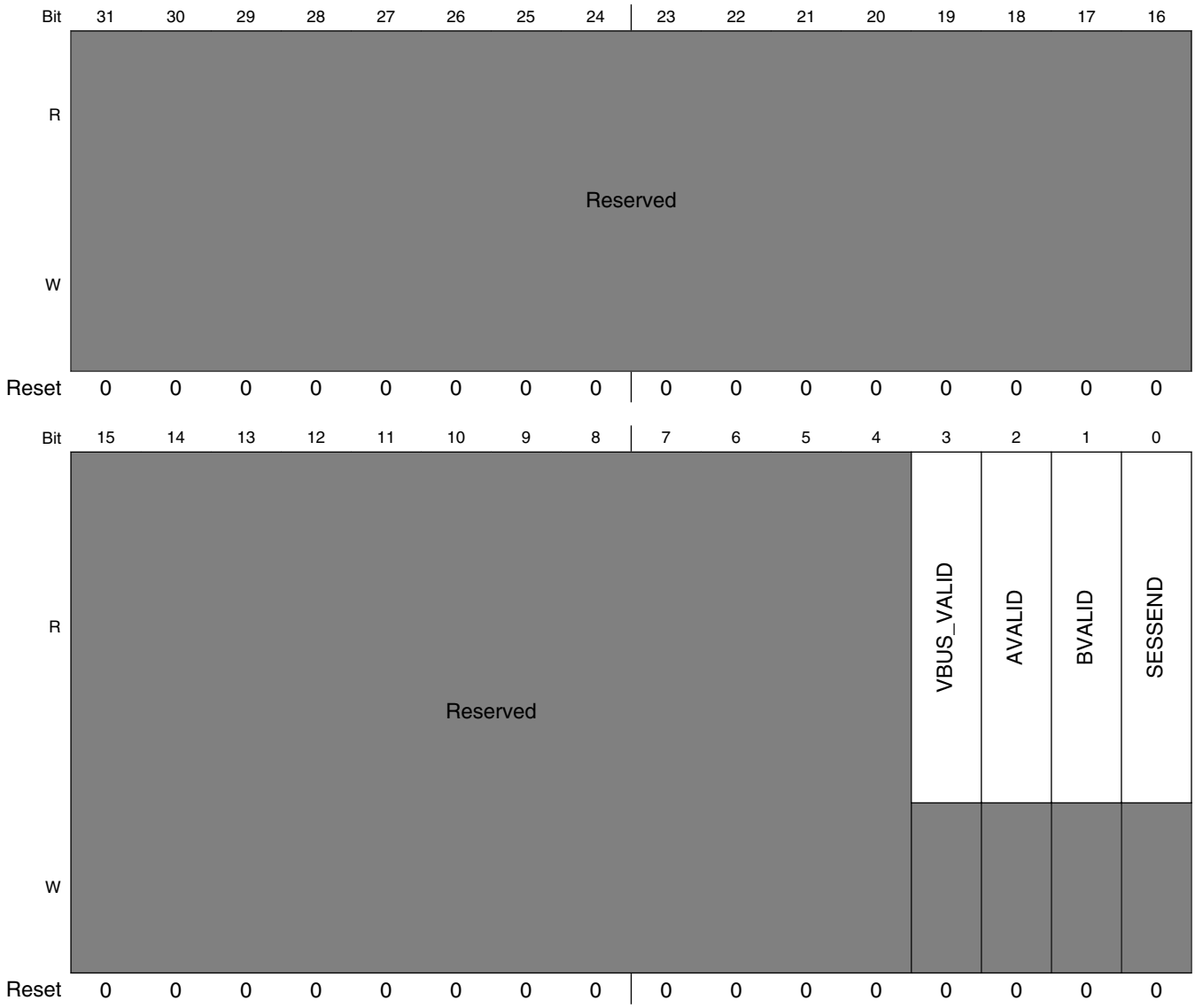
USB_ANALOG_USB2_CHRG_DETECT*n* field descriptions (continued)

Field	Description
18 CHK_CONTACT	0 **NO_CHECK** — Do not check the contact of USB plug. 1 **CHECK** — Check whether the USB plug has been in contact with each other
-	This field is reserved. Reserved.

66.4.8 USB VBUS Detect Status Register (USB_ANALOG_USB2_VBUS_DETECT_STAT)

This register defines fields for USB VBUS Detect status.

Address: 20C_8000h base + 220h offset = 20C_8220h



USB_ANALOG_USB2_VBUS_DETECT_STAT field descriptions

Field	Description
31–4 -	This field is reserved. Reserved.

Table continues on the next page...

USB_ANALOG_USB2_VBUS_DETECT_STAT field descriptions (continued)

Field	Description
3 VBUS_VALID	VBus valid for USB OTG. This bit is a read only version of the state of the analog signal. It can not be overwritten by software.
2 AVALID	Indicates VBus is valid for a A-peripheral. This bit is a read only version of the state of the analog signal. It can not be overritten by software.
1 BVALID	Indicates VBus is valid for a B-peripheral. This bit is a read only version of the state of the analog signal. It can not be overritten by software.
0 SESSEND	Session End for USB OTG. This bit is a read only version of the state of the analog signal. It can not be overwritten by software like the SESSEND bit below. NOTE: This bit's default value depends on whether VDD5V is present, 0 if VDD5V is present, 1 if VDD5V is not present.

66.4.9 USB Charger Detect Status Register (USB_ANALOG_USB2_CHRG_DETECT_STAT)

This register defines fields for USB charger detect status.

Address: 20C_8000h base + 230h offset = 20C_8230h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved												DP_STATE	DM_STATE	CHRG_DETECTED	PLUG_CONTACT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USB_ANALOG_USB2_CHRG_DETECT_STAT field descriptions

Field	Description
31–4 -	This field is reserved. Reserved.
3 DP_STATE	DP line state output of the charger detector.
2 DM_STATE	DM line state output of the charger detector.
1 CHRG_ DETECTED	State of charger detection. This bit is a read only version of the state of the analog signal. 0 **CHARGER_NOT_PRESENT** — The USB port is not connected to a charger. 1 **CHARGER_PRESENT** — A charger (either a dedicated charger or a host charger) is connected to the USB port.
0 PLUG_ CONTACT	State of the USB plug contact detector. 0 **NO_CONTACT** — The USB plug has not made contact. 1 **GOOD_CONTACT** — The USB plug has made good contact.

66.4.10 USB Misc Register (USB_ANALOG_USB2_MISC*n*)

This register defines controls for USB.

Address: 20C_8000h base + 250h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved	EN_CLK_UTMI	Reserved													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved														EN_DEGLITCH	HS_USE_EXTERNAL_R
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

USB_ANALOG_USB2_MISC*n* field descriptions

Field	Description
31 -	This field is reserved. Reserved.
30 EN_CLK_UTMI	Enables the clk to the UTMI block.
29–2 -	This field is reserved. Reserved.
1 EN_DEGLITCH	Enable the deglitching circuit of the USB PLL output.
0 HS_USE_ EXTERNAL_R	Use external resistor to generate the current bias for the high speed transmitter. This bit should not be changed unless recommended by Freescale.

66.4.11 Chip Silicon Version (USB_ANALOG_DIGPROG)

The DIGPROG register returns the digital program ID for the silicon.

Address: 20C_8000h base + 260h offset = 20C_8260h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								MAJOR_UPPER								MAJOR_LOWER								MINOR							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USB_ANALOG_DIGPROG field descriptions

Field	Description
31–24 -	This field is reserved. Reserved.
23–16 MAJOR_UPPER	MAJOR upper byte-Read-only value representing the chip type. 0x63 i.MX 6Dual/6Quad
15–8 MAJOR_LOWER	MAJOR lower byte - Read-only value representing a major silicon revision. 0x00 silicon revision 1.x 0x01 silicon revision 2.x
MINOR	MINOR lower byte - Read-only value representing a minor silicon revision. 0x00 silicon revision x.1 0x01 silicon revision x.2

Chapter 67
Ultra Secured Digital Host Controller (uSDHC)

67.1 Overview

The Ultra Secured Digital Host Controller (uSDHC) provides the interface between the host system and the SD/SDIO/MMC cards, as depicted in Figure 67-1.

The uSDHC acts as a bridge, passing host bus transactions to the SD/SDIO/MMC cards by sending commands and performing data accesses to/from the cards.

It handles the SD/SDIO/MMC protocols at the transmission level.

The following are brief descriptions of the cards supported by the uSDHC:

The Multi Media Card (MMC) is a universal low cost data storage and communication media designed to cover a wide array of applications including mobile video and gaming. Previous MMC cards were based on a 7-pin serial bus with a single data pin, while the new high speed MMC communication is based on an advanced 11-pin serial bus designed to operate in the low voltage range.

The Secure Digital Card (SD) is an evolution of the old MMC technology. It is specifically designed to meet the security, capacity, performance, and environment requirements inherent in newly-emerging audio and video consumer electronic devices. The physical form factor, pin assignment and data transfer protocol are forward-compatible with the old MMC (with some additions).

Under the SD protocol, it can be categorized into Memory card, I/O card and Combo card, which has both memory and I/O functions. The memory card invokes a copyright protection mechanism that complies with the security of the SDMI standard. The I/O card, which is also known as SDIO card, provides high-speed data I/O with low power consumption for mobile electronic devices. For the sake of simplicity, the following figure does not show cards with reduced size or mini cards.

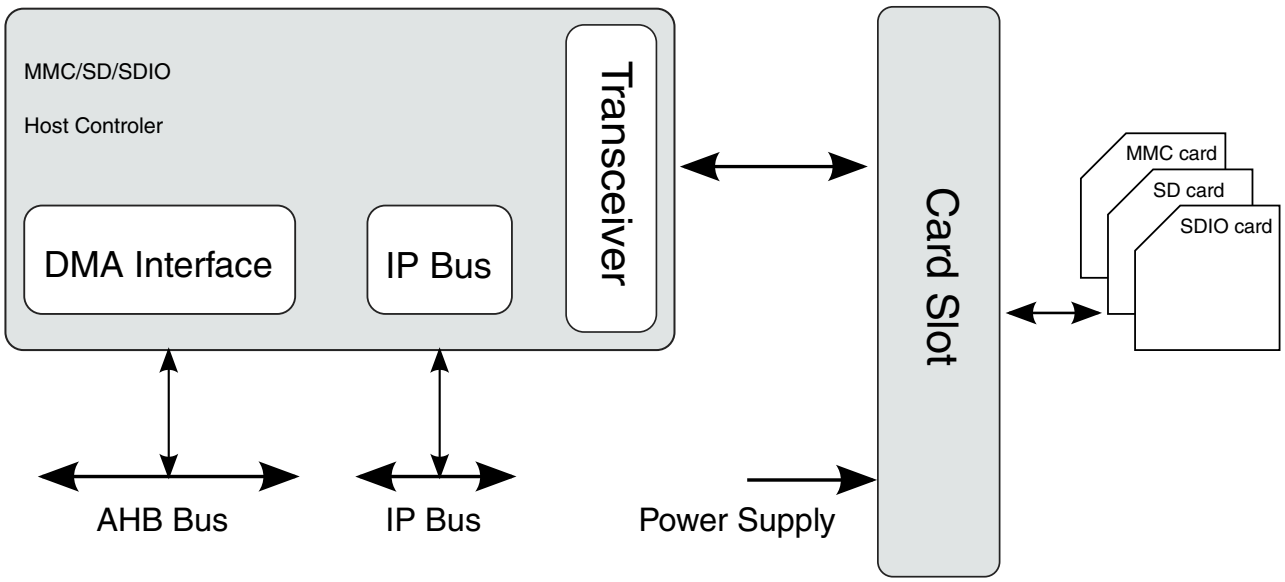


Figure 67-1. System Connection of the uSDHC

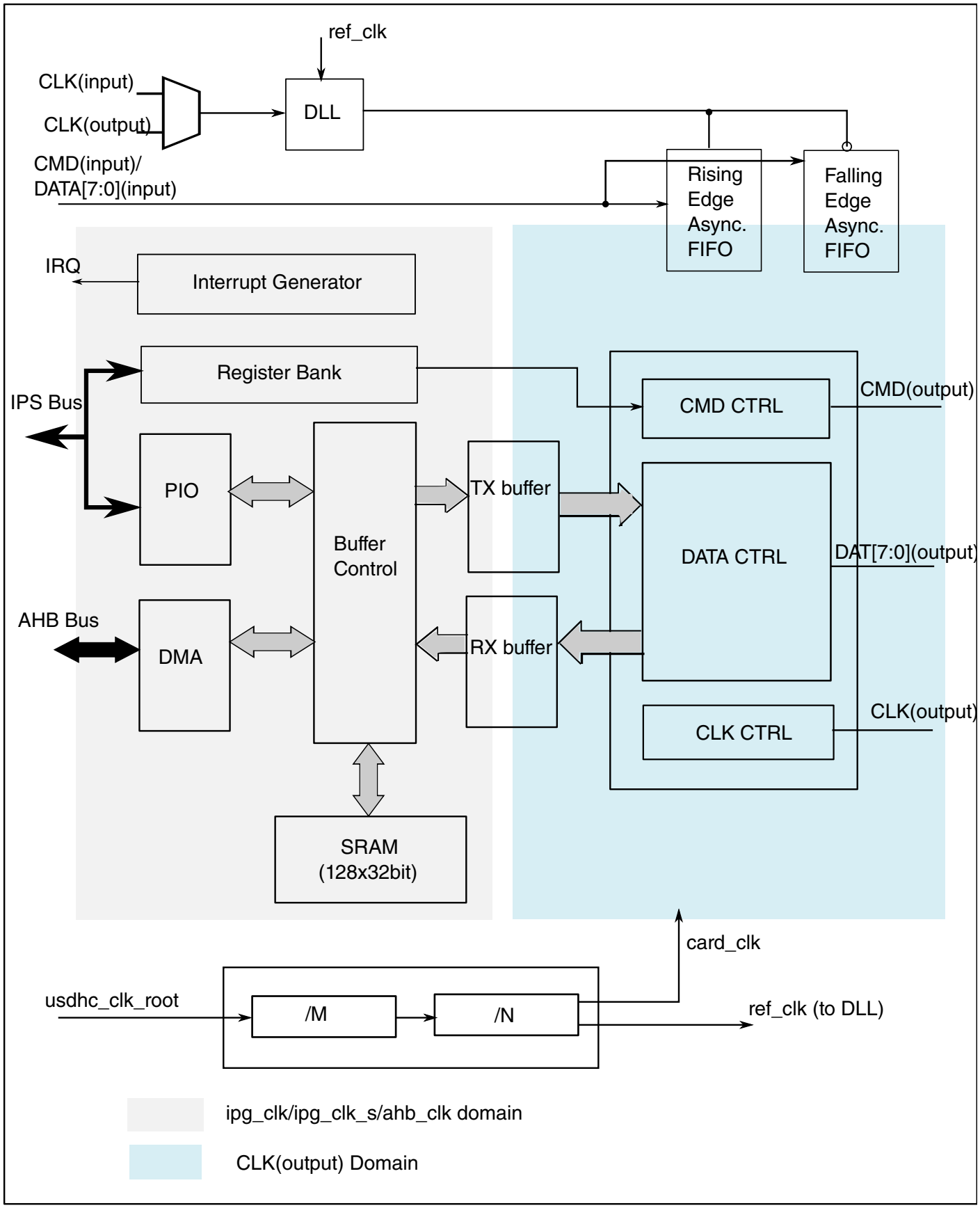


Figure 67-2. ultra Secure Digital Host Controller Block Diagram

67.1.1 Features

The features of the uSDHC module include the following:

- Conforms to the SD Host Controller Standard Specification version 3.0
- Compatible with the MMC System Specification version 4.2/4.3/4.4/4.41
- Compatible with the SD Memory Card Specification version 3.0 and supports the Extended Capacity SD Memory Card
- Compatible with the SDIO Card Specification version 3.0
- Designed to work with SD Memory, miniSD Memory, SDIO, miniSDIO, SD Combo, MMC, MMC plus, and MMC RS cards
- Card bus clock frequency up to 208 MHz
- Supports 1-bit / 4-bit SD and SDIO modes, 1-bit / 4-bit / 8-bit MMC modes
    - Up to 832 Mbps of data transfer for SDIO cards using 4 parallel data lines in SDR(Single Data Rate) mode
    - Up to 400 Mbps of data transfer for SDIO card using 4 parallel data lines in DDR(Dual Data Rate) mode
    - Up to 832 Mbps of data transfer for SDXC cards using 4 parallel data lines in SDR(Single Data Rate) mode
    - Up to 400 Mbps of data transfer for SDXC card using 4 parallel data lines in DDR(Dual Data Rate) mode
    - Up to 416 Mbps of data transfer for MMC cards using 8 parallel data lines in SDR(Single Data Rate) mode
    - Up to 832 Mbps of data transfer for MMC cards using 8 parallel data lines in DDR(Dual Data Rate) mode
- Supports single block/multi-block read and write
- Supports block sizes of 1 ~ 4096 bytes
- Supports the write protection switch for write operations
- Supports both synchronous and asynchronous abort
- Supports pause during the data transfer at block gap
- Supports SDIO Read Wait and Suspend Resume operations
- Supports Auto CMD12 for multi-block transfer
- Host can initiate non-data transfer command while data transfer is in progress
- Allows cards to interrupt the host in 1-bit and 4-bit SDIO modes, also supports interrupt period
- Embodies a fully configurable 128x32-bit FIFO for read/write data
- Supports internal and external DMA capabilities
- Support voltage selection by configuring vendor specific register bit
- Supports Advanced DMA to perform linked memory access

67.1.2 Modes and Operations

67.1.2.1 Data transfer Modes

The uSDHC can select the following modes for data transfer:

- SD 1-bit
- SD 4-bit
- MMC 1-bit
- MMC 4-bit
- MMC 8-bit
- Identification Mode (up to 400 kHz)
- MMC full speed mode (up to 20 MHz)
- MMC high speed mode (up to 52 MHz)
- MMC DDR mode (52MHz both edges)
- SD/SDIO full speed mode (up to 25 MHz)
- SD/SDIO high speed mode (up to 50 MHz)
- SD/SDIO UHS-I mode(up to 208Mhz in SDR mode, up to 50Mhz in DDR mode)

67.2 External Signals

The following table describes the external signals of USDHC:

Table 67-1. USDHC1 External Signals

Signal	Description	Pad	Mode	Direction
SD1_CD_B (CD_B)	Card detection pin If not used(for the embedded memory),tie low to indicate there is a card attached.	GPIO_1	ALT6	I
SD1_CLK (CLK)	Clock for MMC/SD/SDIO card	SD1_CLK	ALT0	O
SD1_CMD (CMD)	CMD line connect to card	SD1_CMD	ALT0	IO
SD1_DATA0 (DATA0)	DATA0 line in all modes Also used to detect busy state	SD1_DAT0	ALT0	IO
SD1_DATA1 (DATA1)	DATA1 line in 4/8-bit mode Also used to detect interrupt in 1/4-bit mode	SD1_DAT1	ALT0	IO
SD1_DATA2 (DATA2)	DATA2 line or Read Wait in 4-bit mode	SD1_DAT2	ALT0	IO

Table continues on the next page...

Table 67-1. USDHC1 External Signals (continued)

Signal	Description	Pad	Mode	Direction
	Read Wait in 1-bit mode			
SD1_DATA3 (DATA3)	DATA3 line in 4/8-bit mode or configured as card detection pin May be configured as card detection pin in 1-bit mode	SD1_DAT3	ALT0	IO
SD1_DATA4 (DATA4)	DATA4 line in 8-bit mode, not used in other modes	NANDF_D0	ALT1	IO
SD1_DATA5 (DATA5)	DATA5 line in 8-bit mode, not used in other modes	NANDF_D1	ALT1	IO
SD1_DATA6 (DATA6)	DATA6 line in 8-bit mode, not used in other modes	NANDF_D2	ALT1	IO
SD1_DATA7 (DATA7)	DATA7 line in 8-bit mode, not used in other modes	NANDF_D3	ALT1	IO
SD1_LCTL (LCTL)	LED control used to drive an external LED Active high Fully controlled by the driver Optional output	GPIO_16	ALT3	O
SD1_VSELECT (VSELECT)	IO power voltage selection signal	KEY_COL1	ALT6	O
		KEY_ROW3	ALT6	
SD1_WP (WP)	Card write protect detect If not used(for the embedded memory), tie low to indicate it's not write protected.	DI0_PIN4	ALT3	I
		GPIO_9	ALT6	

Table 67-2. USDHC2 External Signals

Signal	Description	Pad	Mode	Direction
SD2_CD_B (CD_B)	Card detection pin If not used(for the embedded memory),tie low to indicate there is a card attached.	GPIO_4	ALT6	I
SD2_CLK (CLK)	Clock for MMC/SD/SDIO card	SD2_CLK	ALT0	O
SD2_CMD (CMD)	CMD line connect to card	SD2_CMD	ALT0	IO
SD2_DATA0 (DATA0)	DATA0 line in all modes Also used to detect busy state	SD2_DAT0	ALT0	IO
SD2_DATA1 (DATA1)	DATA1 line in 4/8-bit mode Also used to detect interrupt in 1/4-bit mode	SD2_DAT1	ALT0	IO
SD2_DATA2 (DATA2)	DATA2 line or Read Wait in 4-bit mode Read Wait in 1-bit mode	SD2_DAT2	ALT0	IO
SD2_DATA3 (DATA3)	DATA3 line in 4/8-bit mode or configured as card detection pin	SD2_DAT3	ALT0	IO

Table continues on the next page...

Table 67-2. USDHC2 External Signals (continued)

Signal	Description	Pad	Mode	Direction
	May be configured as card detection pin in 1-bit mode			
SD2_DATA4 (DATA4)	DATA4 line in 8-bit mode, not used in other modes	NANDF_D4	ALT1	IO
SD2_DATA5 (DATA5)	DATA5 line in 8-bit mode, not used in other modes	NANDF_D5	ALT1	IO
SD2_DATA6 (DATA6)	DATA6 line in 8-bit mode, not used in other modes	NANDF_D6	ALT1	IO
SD2_DATA7 (DATA7)	DATA7 line in 8-bit mode, not used in other modes	NANDF_D7	ALT1	IO
SD2_LCTL (LCTL)	LED control used to drive an external LED Active high Fully controlled by the driver Optional output	GPIO_6	ALT6	O
SD2_VSELECT (VSELECT)	IO power voltage selection signal	KEY_ROW1	ALT6	O
		KEY_ROW2	ALT4	
SD2_WP (WP)	Card write protect detect If not used(for the embedded memory), tie low to indicate it's not write protected.	GPIO_2	ALT6	I

Table 67-3. USDHC3 External Signals

Signal	Description	Pad	Mode	Direction
SD3_CLK (CLK)	Clock for MMC/SD/SDIO card	SD3_CLK	ALT0	O
SD3_CMD (CMD)	CMD line connect to card	SD3_CMD	ALT0	IO
SD3_DATA0 (DATA0)	DATA0 line in all modes Also used to detect busy state	SD3_DAT0	ALT0	IO
SD3_DATA1 (DATA1)	DATA1 line in 4/8-bit mode Also used to detect interrupt in 1/4-bit mode	SD3_DAT1	ALT0	IO
SD3_DATA2 (DATA2)	DATA2 line or Read Wait in 4-bit mode Read Wait in 1-bit mode	SD3_DAT2	ALT0	IO
SD3_DATA3 (DATA3)	DATA3 line in 4/8-bit mode or configured as card detection pin May be configured as card detection pin in 1-bit mode	SD3_DAT3	ALT0	IO
SD3_DATA4 (DATA4)	DATA4 line in 8-bit mode, not used in other modes	SD3_DAT4	ALT0	IO
SD3_DATA5 (DATA5)	DATA5 line in 8-bit mode, not used in other modes	SD3_DAT5	ALT0	IO
SD3_DATA6 (DATA6)	DATA6 line in 8-bit mode, not used in other modes	SD3_DAT6	ALT0	IO

Table continues on the next page...

Table 67-3. USDHC3 External Signals (continued)

Signal	Description	Pad	Mode	Direction
SD3_DATA7 (DATA7)	DATA7 line in 8-bit mode, not used in other modes	SD3_DAT7	ALT0	IO
SD3_RESET (RESET)	Card hardware reset signal, active LOW	SD3_RST	ALT0	O
SD3_VSELECT (VSELECT)	IO power voltage selection signal	GPIO_18	ALT2	O
		NANDF_CS1	ALT2	

Table 67-4. USDHC4 External Signals

Signal	Description	Pad	Mode	Direction
SD4_CLK (CLK)	Clock for MMC/SD/SDIO card	SD4_CLK	ALT0	O
SD4_CMD (CMD)	CMD line connect to card	SD4_CMD	ALT0	IO
SD4_DATA0 (DATA0)	DATA0 line in all modes Also used to detect busy state	SD4_DAT0	ALT1	IO
SD4_DATA1 (DATA1)	DATA1 line in 4/8-bit mode Also used to detect interrupt in 1/4-bit mode	SD4_DAT1	ALT1	IO
SD4_DATA2 (DATA2)	DATA2 line or Read Wait in 4-bit mode Read Wait in 1-bit mode	SD4_DAT2	ALT1	IO
SD4_DATA3 (DATA3)	DATA3 line in 4/8-bit mode or configured as card detection pin May be configured as card detection pin in 1-bit mode	SD4_DAT3	ALT1	IO
SD4_DATA4 (DATA4)	DATA4 line in 8-bit mode, not used in other modes	SD4_DAT4	ALT1	IO
SD4_DATA5 (DATA5)	DATA5 line in 8-bit mode, not used in other modes	SD4_DAT5	ALT1	IO
SD4_DATA6 (DATA6)	DATA6 line in 8-bit mode, not used in other modes	SD4_DAT6	ALT1	IO
SD4_DATA7 (DATA7)	DATA7 line in 8-bit mode, not used in other modes	SD4_DAT7	ALT1	IO
SD4_RESET (RESET)	Card hardware reset signal, active LOW	NANDF_ALE	ALT1	O
SD4_VSELECT (VSELECT)	IO power voltage selection signal	NANDF_CS1	ALT1	O

67.2.1 Signals Overview

The uSDHC has 14 associated I/O signals.

- The CLK is an internally generated clock used to drive the MMC, SD, SDIO cards.

- The CMD I/O is used to send commands and receive responses to and from the card. Eight data lines (DAT7~DAT0) are used to perform data transfers between the uSDHC and the card.
- The CD and WP are card detection and write protection signals directly routed from the socket. These two signals are active low (0). A low on CD# means that a card is inserted, and a high on WP means that the write protect switch is active.
- LCTL is an output signal used to drive an external LED to indicate that the SD interface is busy.
- RST is an output signal used to reset the MMC card.
- VSELECT is an output signal used to change the voltage of the external power supplier.

CD, WP, LCTL, RST and VSELECT are all optional for system implementation. If the uSDHC needs to support a 4-bit data transfer, DAT7~DAT4 can also be optional and tied to high.

67.3 Clocks

The table found here describes the clock sources for uSDHC.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 67-5. uSDHC Clocks

Clock name	Clock Root	Description
hclk	ahb_clk_root	AHB bus clock
ipg_clk	ipg_clk_root	Peripheral clock
ipg_clk_perclk	usdhc_clk_root	Base clock
ipg_clk_s	ipg_clk_root	Peripheral access clock for register accesses

67.4 Functional Description

The following sections provide a brief functional description of the major system blocks, including the Data Buffer, DMA AHB interface, register bank as well as IP Bus interface, dual-port memory wrapper, data/command controller, clock & reset manager and clock generator.

67.4.1 Data Buffer

The uSDHC uses one configurable data buffer to transfer data between the system bus (IP Bus or AHB Bus) and the SD card in an optimized manner, maximizing throughput between the two clock domains (IP peripheral clock and the master clock).

The buffer is used as temporary storage for data being transferred between the host system and the card. The watermark levels for read and write are both configurable and can be from 1 to 128 words. The burst lengths for read and write are also configurable and can be from 1 to 31 words.

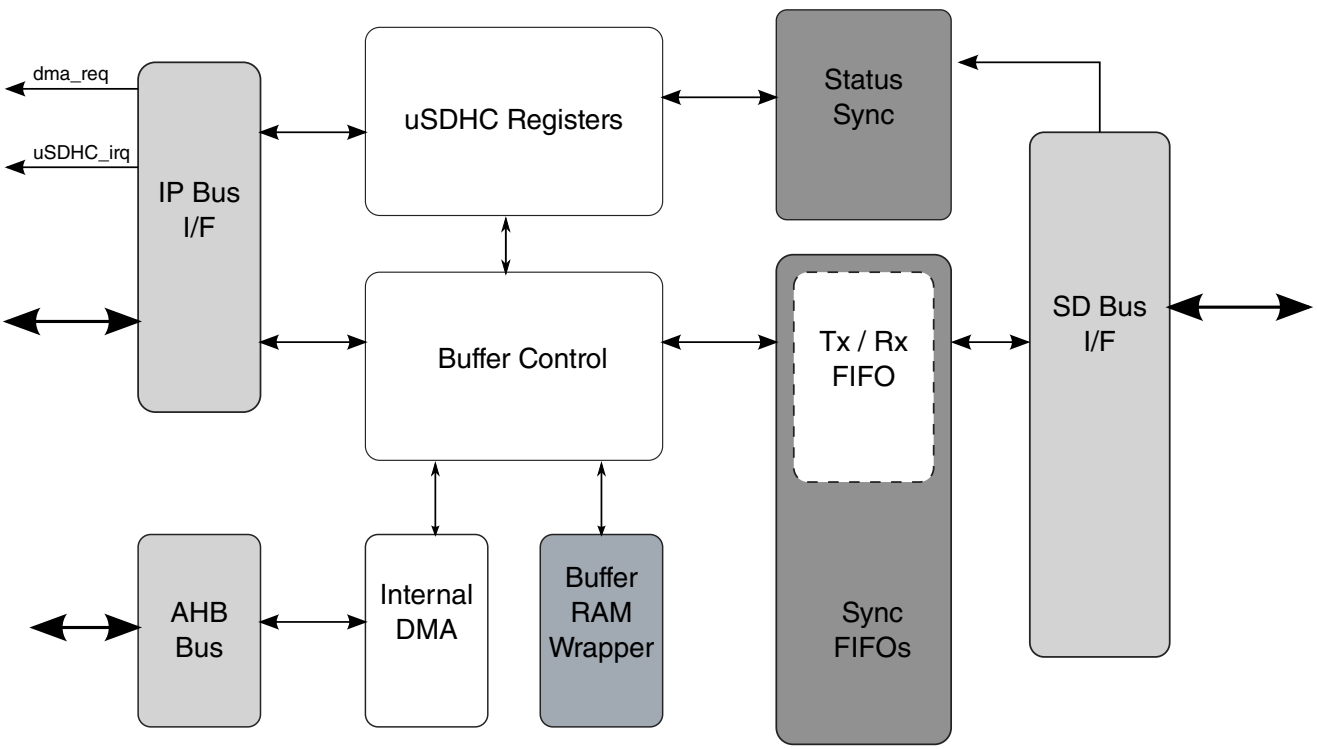


Figure 67-3. uSDHC Buffer Scheme

There are 3 transfer modes to access the data buffer:

- CPU polling mode:
  - For a host read operation, when the number of words received in the buffer meets or exceeds the RD_WML watermark value, by polling the BRR bit, the Host Driver can read the Buffer Data Port register to fetch the amount of words set in the RD_WML register from the buffer. The write operation is similar.
- External DMA mode:
  - For a read operation, when there are more words received in the buffer than the amount set in the RD_WML register, a DMA request is sent out to inform the external DMA to fetch the data. The request will be immediately de-asserted when there is an access on the Buffer Data Port register. If the number of words

in the buffer after the current burst meets or exceeds RD_WML value, the DMA request is asserted again. For instance, if there are twice as many words in the buffer as there are in the RD_WML value, there are two successive DMA requests with only one cycle of de-assertion between. The write operation is similar. Note the accesses CPU polling mode and external DMA mode both use the IP bus, and if the external DMA is enabled, in both modes an external DMA request is sent when the buffer is ready.

- Internal DMA mode (includes simple and advanced DMA accesses):
  - The internal DMA access, either by simple or advanced DMA, is over the AHB bus. For internal DMA access mode, the external DMA request will never be sent out.

For a read operation, when there are more words in the buffer than the amount set in the RD_WML register, the internal DMA starts fetching data over the AHB bus. Except for INCR4 and INCR8, the burst type is always INCR mode and the burst length depends on the shortest of following factors:

- Burst length configured in the burst length field of the Watermark Level register
- Watermark Level boundary
- Block size boundary
- Data boundary configured in the current descriptor (if the ADMA is active)
- 1 Kbyte address boundary defined in the AHB protocol

Write operation is similar.

Sequential and contiguous access is necessary to ensure the pointer address value is correct. Random or skipped access is not possible. The byte order, by reset, is little endian mode. The actual byte order is swapped inside the buffer, according to the endian mode configured by software (see the following figures). For a host write operation, byte order is swapped after data is fetched from the buffer and ready to send to the SD Bus. For a host read operation, byte order is swapped before the data is stored in the buffer.

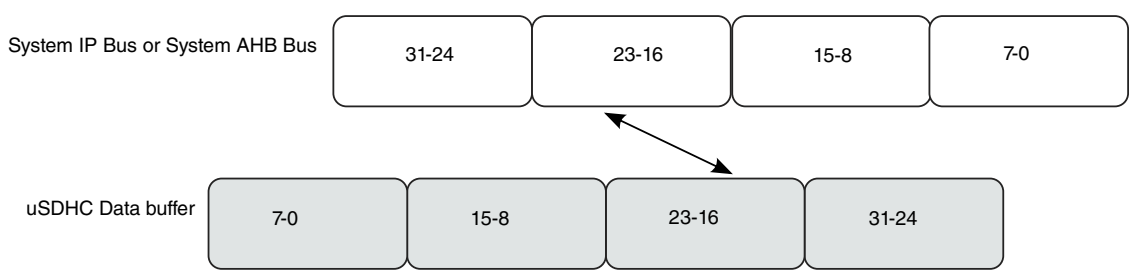


Figure 67-4. Data Swap between System Bus and uSDHC Data Buffer in Byte Little Endian Mode

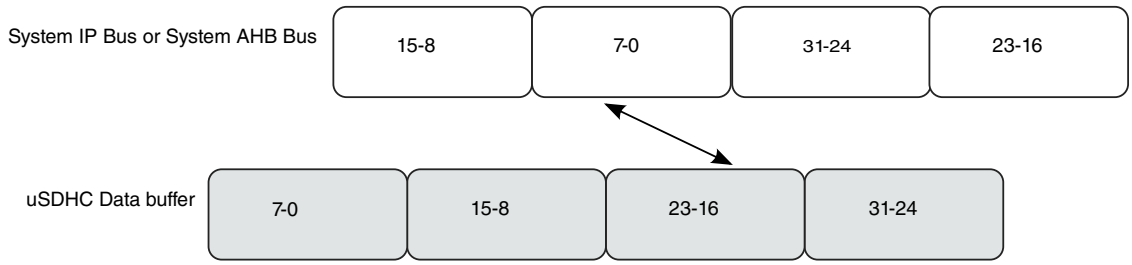


Figure 67-5. Data Swap between System Bus and uSDHC Data Buffer in Half Word Big Endian Mode

67.4.1.1 Write Operation Sequence

There are three ways to write data into the buffer when the user transfers data to the card:

- External DMA through the uSDHC DMA request signal
- Processor core polling through the BWR bit in Interrupt Status register (interrupt or polling)
- Internal DMA

When the internal DMA is not used, (the DMAEN bit in the Transfer Type register is not set when the command is sent), the uSDHC asserts a DMA request when the amount of buffer space exceeds the value set in the WR_WML register, and is ready for receiving new data. At the same time, the uSDHC sets the BWR bit. The buffer write ready interrupt will be generated if it is enabled by software.

When internal DMA is used, the uSDHC will not inform the system before all the required number of bytes are transferred (if no error was encountered). When an error occurs during the data transfer, the uSDHC will abort the data transfer and abandon the current block. The Host Driver should read the contents of the DMA System Address register to obtain the starting address of the abandoned data block. If the current data transfer is in multi-block mode, the uSDHC will not automatically send CMD12, even though the AC12EN bit in the Transfer Type register is set. The Host Driver sends CMD12 in this scenario and re-starts the write operation from that address. It is recommended that a Software Reset for Data be applied before the transfer is re-started.

The uSDHC will not start data transmission until the number of words set in the WR_WML register can be held in the buffer. If the buffer is empty and the Host System does not write data in time, the uSDHC will stop the CLK to avoid the data buffer under-run situation.

67.4.1.2 Read Operation Sequence

There are three ways to read data from the buffer when the user transfers data to the card:

- External DMA through the uSDHC DMA request signal
- Processor core polling through the BRR bit in Interrupt Status register (interrupt or polling)
- Internal DMA

When internal DMA is not used (DMAEN bit in Transfer Type register is not set when the command is sent), the uSDHC asserts a DMA request when the amount of data exceeds the value set in the RD_WML register, that is available and ready for system fetching data. At the same time, the uSDHC sets the BRR bit. The buffer read ready interrupt will be generated if it is enabled by software.

When internal DMA is used, the uSDHC will not inform the system before all the required number of bytes are transferred (if no error was encountered). When an error occurs during the data transfer, the uSDHC will abort the data transfer and abandon the current block. The Host Driver should read the content of the DMA System Address register to get the starting address of the abandoned data block. If the current data transfer is in multi-block mode, the uSDHC will not automatically send CMD12, even though the AC12EN bit in the Transfer Type register is set. The Host Driver sends CMD12 in this scenario and re-starts the read operation from that address. It is recommended that a Software Reset for Data be applied before the transfer is re-started.

For any write transfer mode, the uSDHC will not start data transmission until the number of words set in the RD_WML register are in the buffer. If the buffer is full and the Host System does not read data in time, the uSDHC will stop the CLK to avoid the data buffer over-run situation.

67.4.1.3 Data Buffer and Block Size

The user needs to know the buffer size for the buffer operation during a data transfer to utilize it in the most optimized way. In the uSDHC, the only data buffer can hold up to 128 words (32-bit) and the watermark levels for write and read can be configured accordingly.

For both read and write, the watermark level can be from 1 to 128 words. For both read and write the burst length can be from 1 to 31 words. The Host Driver may configure the value according to the system situation and requirement.

During a multi-block data transfer, the block length can be set to any value between 1 and 4096 bytes, satisfying the requirements of the external card. The only restriction is from the external card, which can be limited in size or support of a partial block access (which is not the integer times of 512 bytes).

As uSDHC treats each block individually, for block sizes which are not multiples of four (not word-aligned) stuffed bytes are required at the end of each block. For example, if the block size is 7 bytes and there are 12 blocks to write, the system side must write two times for each block. For each block the ending byte will be abandoned by uSDHC because it only sends 7 bytes to the card and picks data from the following system write, resulting in 24 beats of write access in total.

67.4.1.4 Dividing Large Data Transfer

This SDIO command CMD53 definition limits the maximum data size of data transfers according to the following formula:

Max data size = Block size x Block count

The length of a multiple block transfer needs to be in block size units. If the total data length can't be divided evenly into a multiple of the block size, then there are two ways to transfer the data which depend on the function and the card design. Option 1 is for the Host Driver to split the transaction. The remainder of the block size data is then transferred by using a single block command at the end. Option 2 is to add dummy data in the last block to fill the block size. For option 2, the card must manage the removal of the dummy data.

See the figure below for an example showing the dividing of large data transfers, assuming a kind of WLAN SDIO card that only supports a block size up to 64 bytes. Although the uSDHC supports a block size of up to 4096 bytes, the SDIO can only accept a block size less than 64 bytes, so the data must be divided (see example below).

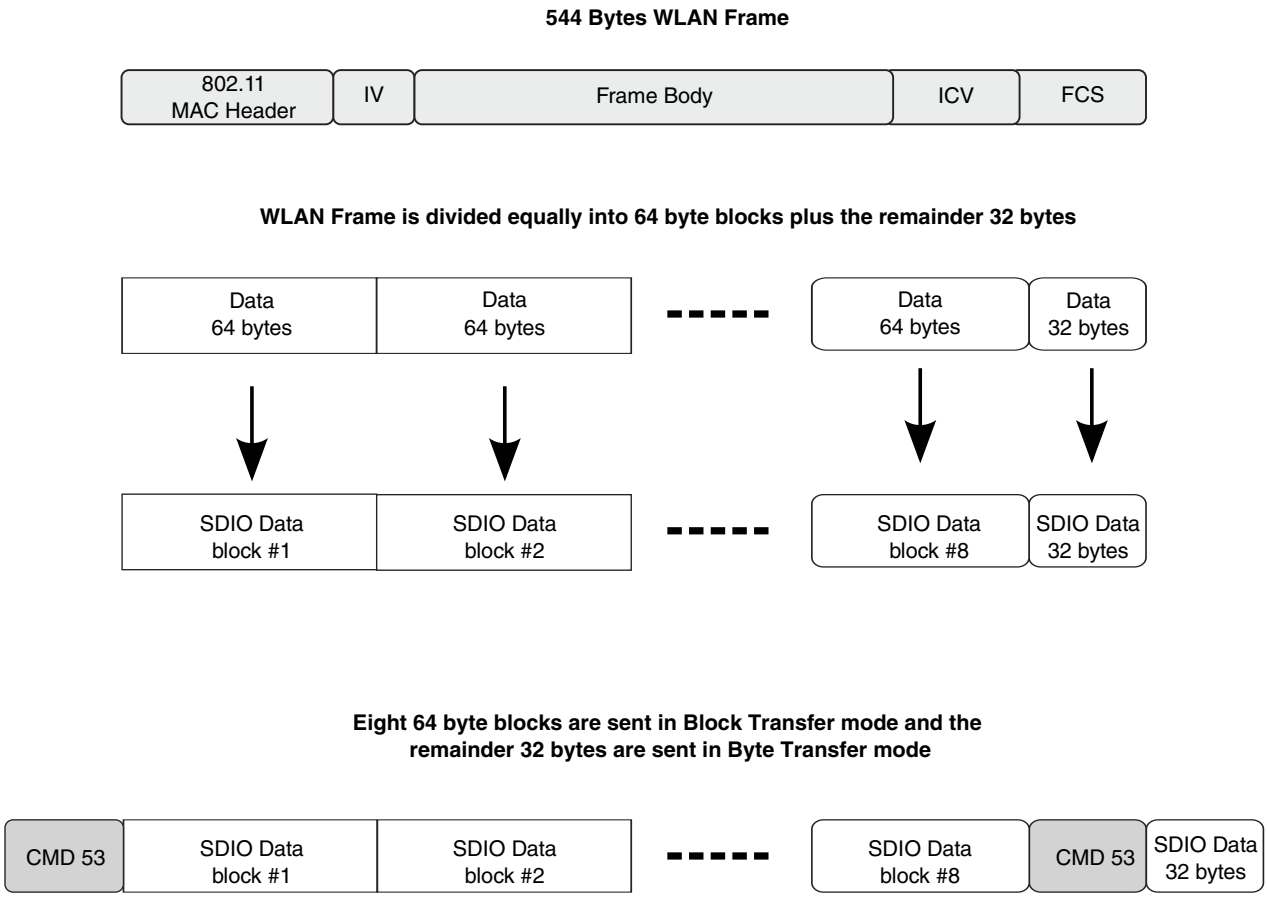


Figure 67-6. Example for Dividing Large Data Transfers

67.4.1.5 External DMA Request

When the internal DMA is not in use and external DMA is enabled, the Data Buffer will generate a DMA request to the system. During a write operation, when the number of WR_WML words can be held in the buffer free space, the signal uSDHC_dreq_b is asserted to 0, informing the Host System of a DMA write.

The BWR bit in the Interrupt Status register is also set, as long as the BWRSEN bit in the Interrupt Status Enable register is set. The DMA request is de-asserted after several accesses to the Data Port register are made while the buffer's free space can't meet the watermark condition (free space> write watermark level).

On read operation, when the number of RD_WML words are already in the buffer, the signal uSDHC_dreq_b is asserted to 0, informing the Host System for a DMA read. The BRR bit in the Interrupt Status register is also set, as long as the BRRSEN bit in the

Interrupt Status Enable register is set. The DMA request is de-asserted after several accesses to the Data Port register are made while the buffer's data can't meet the watermark condition (the number of data in buffer > read watermark level).

If the DMA burst length can't change during a data transfer for an external DMA transfer, the watermark level (read or write) must be a divisor of the block size. If it is not, transferring the block may cause buffer under-run (read operation) or over-run (write operation). For example, if the block size is 512 bytes, the watermark level of read (or write) must be a power of two between 1 and 128. For processor core polling access there is no such issue, as the last access in the block transfer can be controlled by software. The watermark level can be any value, even larger than the block size (but no greater than 128 words) because the actual number of bytes transferred by the software can be controlled and does not exceed the block size in each transfer.

The uSDHC also supports non-word aligned block size, as long as the card supports that block size. In this case, the watermark level should be set as the number of words. For example, if the block size is 31 bytes, the watermark level can be set to any number of words. For this case, the BLKSIZE bits of the Block Attribute register will be set as 1fh. For the CPU polling access, the burst length can be 1 to 128 words, without restriction. This is because the software will transfer 8 words, and the uSDHC will also set the BWR or BRR bits when the remaining data does not violate data buffer. See DMA Burst Length for more details about the dynamic watermark level of the data buffer. For the above example, even though 8 words are transferred via the Data Port register, the uSDHC will transfer only 31 bytes over the SD Bus, as required by the BLKSIZE bits. In this data transfer, with non-word aligned block size, the endian mode should be set cautiously or invalid data will be transferred to and from the card.

67.4.2 DMA AHB Interface

The internal DMA implements a DMA engine and the AHB master. When the internal DMA is enabled, the uSDHC_dreq_b will not be asserted during the transfer, but the BWR and BRR bits will be set if the BWRSEN and BRRSEN bits have been set in the Interrupt Status Enable register.

See the figure below for an illustration of the DMA AHB interface block.

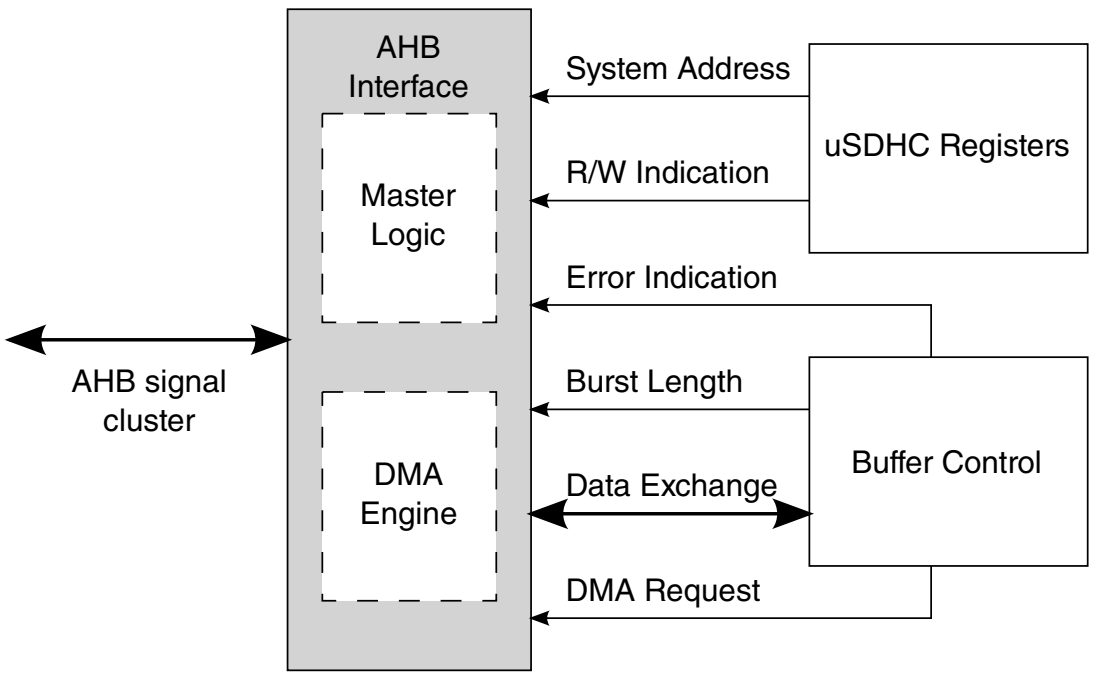


Figure 67-7. DMA AHB Interface Block

67.4.2.1 Internal DMA Request

If the watermark level requirement is met in data transfer or if the last data of current block is ready in the data buffer, and the Internal DMA is enabled, the Data Buffer block will send a DMA request to AHB interface. Meanwhile, the external DMA request signal (uSDHC_dreq_b) is disabled.

The delay in response from the internal DMA engine depends on the system AHB bus loading and the priority assigned to the uSDHC. The DMA engine does not respond to the request during its burst transfer, but is ready to serve as soon as the burst is over. The Data Buffer de-asserts the request if the data buffer space(for write) or bytes in data buffer is smaller than the watermark level. Upon access to the buffer by internal DMA,

the Data Buffer updates its internal buffer pointer, and when the watermark level is satisfied or the last data of current block is ready in the data buffer, another DMA request is sent.

The data transfer is in the block unit, and the subsequent watermark level is always set as the remaining number of words. For instance, for a multi block data read with each block size of 31 bytes, and the burst length set to 6 words. After the first burst transfer, if there are more than 2 words in the buffer (which might contain some data of the next block), another DMA request is sent. This is because the remaining number of words to send for the current block is (31 - 6 * 4) / 4 = 2. The uSDHC will read 2 words out of the buffer, with 7 valid bytes and 1 stuffed byte.

67.4.2.2 DMA Burst Length

Just like a CPU polling access, the DMA burst length for the internal DMA engine can be from 1 to 16 words. The actual burst length for the DMA depends on the lesser of the configured burst length or the remaining words of the current block.

See the example in Internal DMA Request. After 6 words are read, the burst length will be 2 words, then the next burst length will be 6 words. This is because the next block starts, which is 31 bytes, more than 6 words. The Host Driver may take this variable burst length into account. It is also acceptable to configure the burst length as the divisor of the block size, so that each time the burst length will be the same.

67.4.2.3 AHB Master Interface

It is possible that the internal AHB DMA engine could fail during the data transfer. Upon detection of an AHB bus error during DMA transfer, the DMA engine stops the transfer and goes to the idle state.At that point, the internal data buffer stops receiving incoming data and sending out data. The DMAE bit in the Interrupt Status register will be generated to host CPU to report a bus error condition.

Once the DMAE interrupt is received, the software shall send a CMD12 to abort the current transfer and read the DS_ADDR bits of the DMA System Address register to get the starting address of the corrupted block. After the DMA error is fixed, the software should apply a data reset and re-start the transfer from this address to recover the corrupted block. DMA operation will resume when the interrupt is serviced by software.

67.4.2.4 ADMA Engine

In the SD Host Controller Standard, a new DMA transfer algorithm called the ADMA (Advanced DMA) is defined. For Simple DMA, once the page boundary is reached, a DMA interrupt will be generated and the new system address shall be programmed by the Host Driver.

The ADMA defines the programmable descriptor table in the system memory. The Host Driver can calculate the system address at the page boundary and program the descriptor table before executing ADMA. It reduces the frequency of interrupts to the host system. Therefore, higher speed DMA transfers could be realized since the Host MCU intervention would not be needed during long DMA based data transfers.

There are two types of ADMA: ADMA1 and ADMA2 in Host Controller. ADMA1 can support data transfer of 4KB aligned data in system memory. ADMA2 improves the restriction so that data of any location and any size can be transferred in system memory. Their formats of Descriptor Table are different.

ADMA can recognize all kinds of descriptors define in SD Host Controller Standard, and if 'End' flag is detected in the descriptor, ADMA will stop after this descriptor is processed.

67.4.2.4.1 ADMA Concept and Descriptor Format

For ADMA1, including the following descriptors:

- Valid/Invalid descriptor.
- Nop descriptor.
- Set data length descriptor.
- Set data address descriptor.
- Link descriptor.
- Interrupt flag and End flag in descriptor.

For ADMA2, including the following descriptors:

- Valid/Invalid descriptor.
- Nop descriptor.
- Rsv descriptor.
- Set data length & address descriptor.
- Link descriptor.
- Interrupt flag and End flag in descriptor.

See Figure 67-8 for the format of the descriptor table for ADMA1.

ADMA2 deals with the lower 32-bit first, and then the higher 32-bit. If the 'Valid' flag of descriptor is 0, it will ignore the high 32-bit. Address field shall be set on word aligned(lower 2-bit is always set to 0). Data length is in byte unit.

ADMA will start read/write operation after it reaches the Tran state, using the data length and data address analyzed from most recent descriptor(s).

For ADMA1, the valid data length descriptor is the last Set type descriptor before Tran type descriptor. Every Tran type will trigger a transfer, and the transfer data length is extracted from the most recent Set type descriptor. If there is no Set type descriptor after the previous Trans descriptor, the data length will be the value for previous transfer, or 0 if no Set descriptor is ever met.

For ADMA2, Tran type descriptor contains both data length and transfer data address, so only a Tran type descriptor can start a data transfer

Address/ Page Field		Address/ Page Field		Attribute Field					
31	12	11	6	5	4	3	2	1	0
Address or Data Length		000000		Act 2	Act 1	0	Int	End	Valid

Act 2	Act1	Symbol	Comment	31- 28	27- 12
0	0	Nop	No Operation	Don't Care	
0	1	Set	Set Data Length	0000	Data Length
1	0	Tran	Transfer Data	Data Address	
1	1	Link	Link Descriptor	Descriptor Address	

Valid	Valid = 1 indicates this line of descriptor is effective. If Valid = 0 generate ADMA Error Interrupt and stop ADMA.
End	End = 1 indicates current descriptor is the ending one.
Int	Int = 1 generates DMA Interrupt when this descriptor is processed.

Figure 67-8. Format of the ADMA1 Descriptor Table

67.4.2.4.2 ADMA Interrupt

If the interrupt flag descriptor is set, ADMA will generate an interrupt according to various types of descriptors:

For ADMA1:

- Set type of descriptor: interrupt is generated when data length is set.
- Tran type descriptor: interrupt is generated when this transfer is complete.
- Link type of descriptor: interrupt is generated when new descriptor address is set.
- Nop type of descriptor: interrupt is generated just after this descriptor is fetched.

For ADMA2:

- Tran type of descriptor: interrupt is generated when this transfer is complete.
- Link type of descriptor: interrupt is generated when new descriptor address is set.
- Nop/Rsv type of descriptor: interrupt is generated just after this descriptor is fetched.

67.4.2.4.3 ADMA Error

The ADMA will stop whenever any error is encountered. These errors include:

- Fetching descriptor error
- AHB response error
- Data length mismatch error

An ADMA descriptor error will be generated when it fails to detect a 'Valid' flag in the descriptor. If an ADMA descriptor error occurs, the interrupt is not generated even if the 'Interrupt' flag of this descriptor is set.

When BLKCNTEN bit is set, data length set in buffer must be equal to the whole data length set in descriptor nodes, otherwise data length mismatch error will be generated.

When BLKCNTEN bit is not set, then whole data length set in descriptor should be a multiple of block lengths; otherwise, when data set in the descriptor nodes are not performed at block boundaries, then data mismatch errors will occur.

67.4.3 Register Bank with IP Bus Interface

Register accesses via the IP Bus interface are actually on the Register Bank.

See Figure 67-9 below for the block diagram.

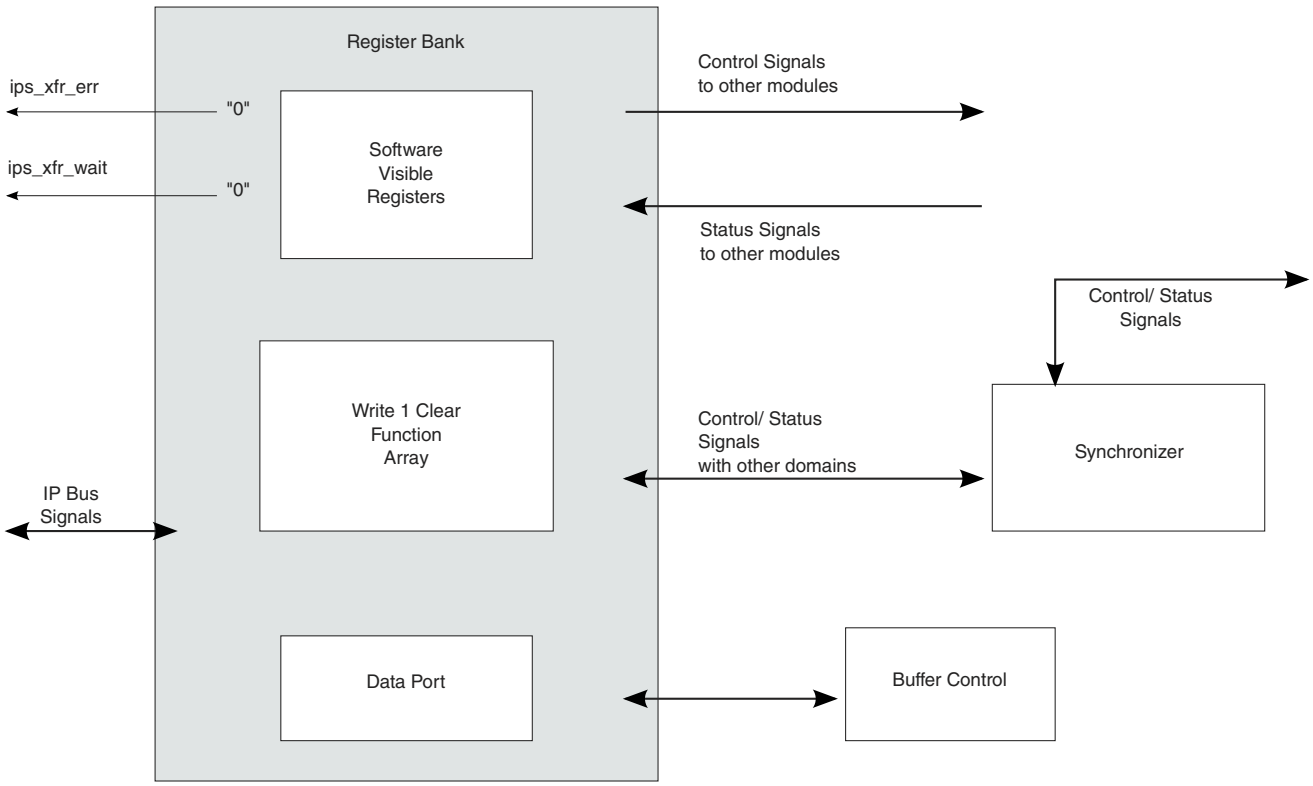


Figure 67-9. Register Bank Diagram

Only 32-bit access is allowed, and no partial read / write is supported, thus all accesses are word aligned.

67.4.3.1 SD Protocol Unit

The SD protocol unit deals with all SD protocol affairs.

The SD Protocol Unit performs the following functions:

- Acts as the bridge between the internal buffer and the SD bus
- Sends the command data as well as its argument serially
- Stores the serial response bit stream into corresponding registers
- Detects the bus state on the CMD/DAT lines
- Monitors the interrupt from the SDIO card
- Asserts the read wait signal
- Gates off the SD clock when buffer is announcing danger status
- Detects the write protect state

The SD Protocol Unit consists of four sub modules:

1. SD control misc.

2. Command control.
3. Data control.
4. Clock control

67.4.3.2 SD control misc

In the SD control misc unit, the card detect(include the CD_B and DATA3 used as Card Detection), write protection and card interrupt are implemented.

This module monitors the signal level on all 8 data lines, the command lines, and directly routes the level values into the Register Bank. The driver can use this for debug purposes.

The module also detects the WP (Write Protect) line. If WP is active, writes to the register bank will be ignored.

This module also drives the LCTL output signal when the LCTL bit is set by the driver.

67.4.3.3 SD Clock control

If the internal data buffer is near full(for read) or near empty(for write), the SD clock must be gated off to avoid buffer over/under-run, this module will assert the gate of the output SD clock to shut the clock off. After the buffer has space(for read) or has data(for write), the clock gate of this module will open and the SD clock will be active again.

67.4.3.4 Command control

The Command Control module deals with the transactions on the CMD line.

See the figure below for an illustration of the structure for the Command CRC Shift Register.

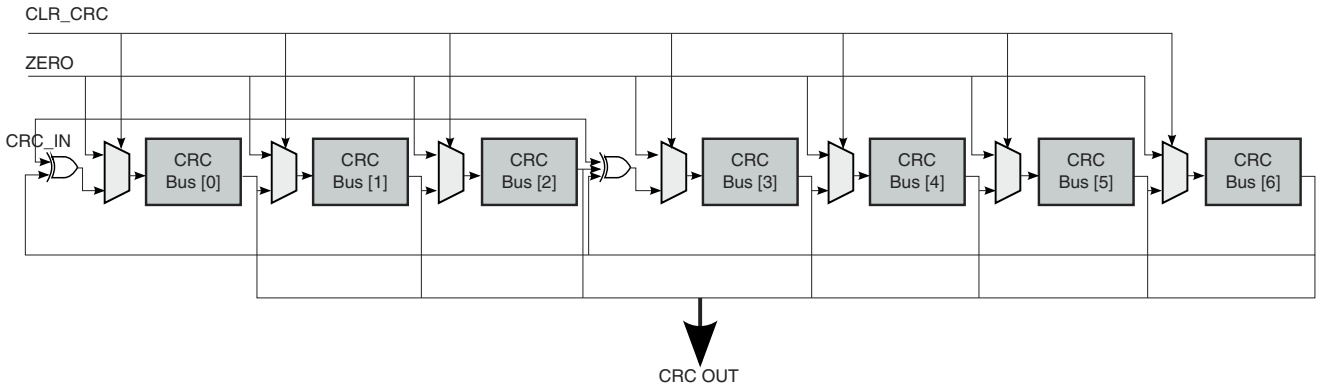


Figure 67-10. Command CRC Shift Register

The CRC polynomials for the CMD are as follows:

```
Generator polynomial: G(x) = x^7 + x^3 + 1
M(x) = (first bit) * x^n + (second bit) * x^n-1 +...+ (last bit) * x^0
CRC[6:0] = Remainder [(M(x) * x^7) / G(x)]
```

67.4.3.5 Data control

The Data Agent deals with the transactions on the eight data lines. Moreover, this module also detects the busy state on the DATA0 line, and generates the Read Wait state by the request from the Transceiver.

The CRC polynomials for the DATA are as follows:

```
Generator polynomial: G(x) = x^16 + x^12 + x^5 +1
M(x) = (first bit) * x^n + (second bit) * x^n-1 +...+ (last bit) * x^0
CRC[15:0] = Remainder [(M(x) * x^16) / G(x)]
```

67.4.4 Clock & Reset Manager

This module controls all the reset signals within the uSDHC.

There are four kinds of reset signals within uSDHC:

1. Hardware reset.
2. Software reset for all logic.
3. Software reset for the data logic.
4. Software reset for the command logic.

All these signals are fed into this module and stable signals are generated inside the module to reset all other modules. The module also gates off all the inside signals.

67.4.5 Clock Generator

The Clock Generator generates the card CLK by peripheral source clock in two stages.

Refer to the figure below for the structure of the divider. The term "Base" represents the frequency of peripheral source clock.

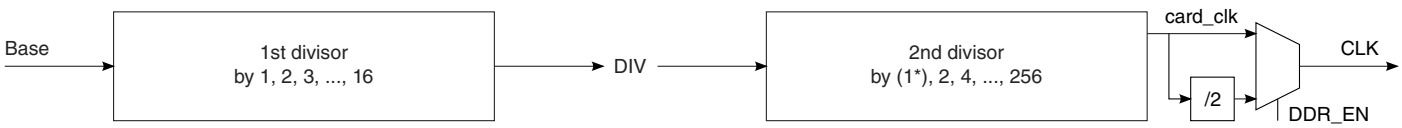


Figure 67-11. Two Stages of the Clock Divider

The first stage outputs an intermediate clock (DIV), which can be Base, Base/2, Base/3, ..., or Base/16.

The second stage is a prescaler, and outputs the actual internal working clock (card_clk). This clock is the driving clock for all sub modules of the SD Protocol Unit, and the sync FIFOs (see Figure 67-3) to synchronize with the data rate from the internal data buffer. The frequency of the clock output from this stage, can be DIV, DIV/2, DIV/4,..., or DIV/256. Thus the highest frequency of the card_clk is Base, and the next highest is Base/2, while the lowest frequency is Base/4096. If the duty cycle of Base clock is 50%, the duty cycle of card_clk is also 50%, even when the compound divisor is an odd value.

Please note, in SDR mode and DDR mode, the CLK are different.

- In SDR mode, CLK is equal to the internal working clock(card_clk).

- In DDR mode, CLK is equal to the card_clk/2.

67.4.6 SDIO Card Interrupt

67.4.6.1 Interrupts in 1-bit Mode

In this case the DATA1 pin is dedicated to providing the interrupt function. An interrupt is asserted by pulling the DATA1 low from the SDIO card, until the interrupt service is finished to clear the interrupt.

67.4.6.2 Interrupt in 4-bit Mode

Since the interrupt and data line 1 share Pin 8 in 4-bit mode, an interrupt will only be sent by the card and recognized by the host during a specific time. This is known as the Interrupt Period. The uSDHC will only sample the level on Pin 8 during the Interrupt Period. At all other times, the host will ignore the level on Pin 8, and treat it as the data signal. The definition of the Interrupt Period is different for operations with single block and multiple block data transfers.

In the case of normal single data block transmissions, the Interrupt Period becomes active two clock cycles after the completion of a data packet. This Interrupt Period lasts until after the card receives the end bit of the next command that has a data block transfer associated with it.

For multiple block data transfers in 4-bit mode, there is only a limited period of time that the Interrupt Period can be active due to the limited period of data line availability between the multiple blocks of data. This requires a more strict definition of the Interrupt Period. For this case, the Interrupt Period is limited to two clock cycles. This begins two clocks after the end bit of the previous data block. During this 2-clock cycle interrupt period, if an interrupt is pending, the DATA1 line will be held low for one clock cycle with the last clock cycle pulling DATA1 high. On completion of the Interrupt Period, the card releases the DATA1 line into the high Z state. The uSDHC samples the DATA1 during the Interrupt Period when the IABG bit in the Protocol Control register is set.

Refer to SDIO Card Specification v1.10f for further information about the SDIO card interrupt.

67.4.6.3 Card Interrupt Handling

When the CINTIEN bit in the Interrupt Signal Enable Register is set to 0, the uSDHC clears the interrupt request to the Host System. The Host Driver should clear this bit before servicing the SDIO Interrupt and should set this bit again after all interrupt requests from the card are cleared to prevent inadvertent interrupts.

The SDIO Card Interrupt Status can be cleared by writing 1 to this bit. But as the interrupt source from the SDIO card does not clear, this bit is set again. In order to clear this bit, it is required to reset the interrupt source from the external card followed by a writing 1 to this bit. In 1-bit mode, the uSDHC will detect the SDIO Interrupt with or without the SD clock (to support wakeup). In 4-bit mode, the interrupt signal is sampled during the Interrupt Period, so there are some sample delays between the interrupt signal from the SDIO card and the interrupt to the Host System Interrupt Controller. When the SDIO status has been set, and the Host Driver needs to service this interrupt, so the SDIO bit in the Interrupt Control Register of SDIO card will be cleared. This is required to clear

the SDIO interrupt status latched in the uSDHC and to stop driving the interrupt signal to the System Interrupt Controller. The Host Driver must issue a CMD52 to clear the card interrupt. After completion of the card interrupt service, the SDIO Interrupt Status Enable bit is set to 1, and the uSDHC starts sampling the interrupt signal again.

See the figure below for an illustration of the SDIO card interrupt scheme and for the sequences of software and hardware events that take place during a card interrupt handling procedure.

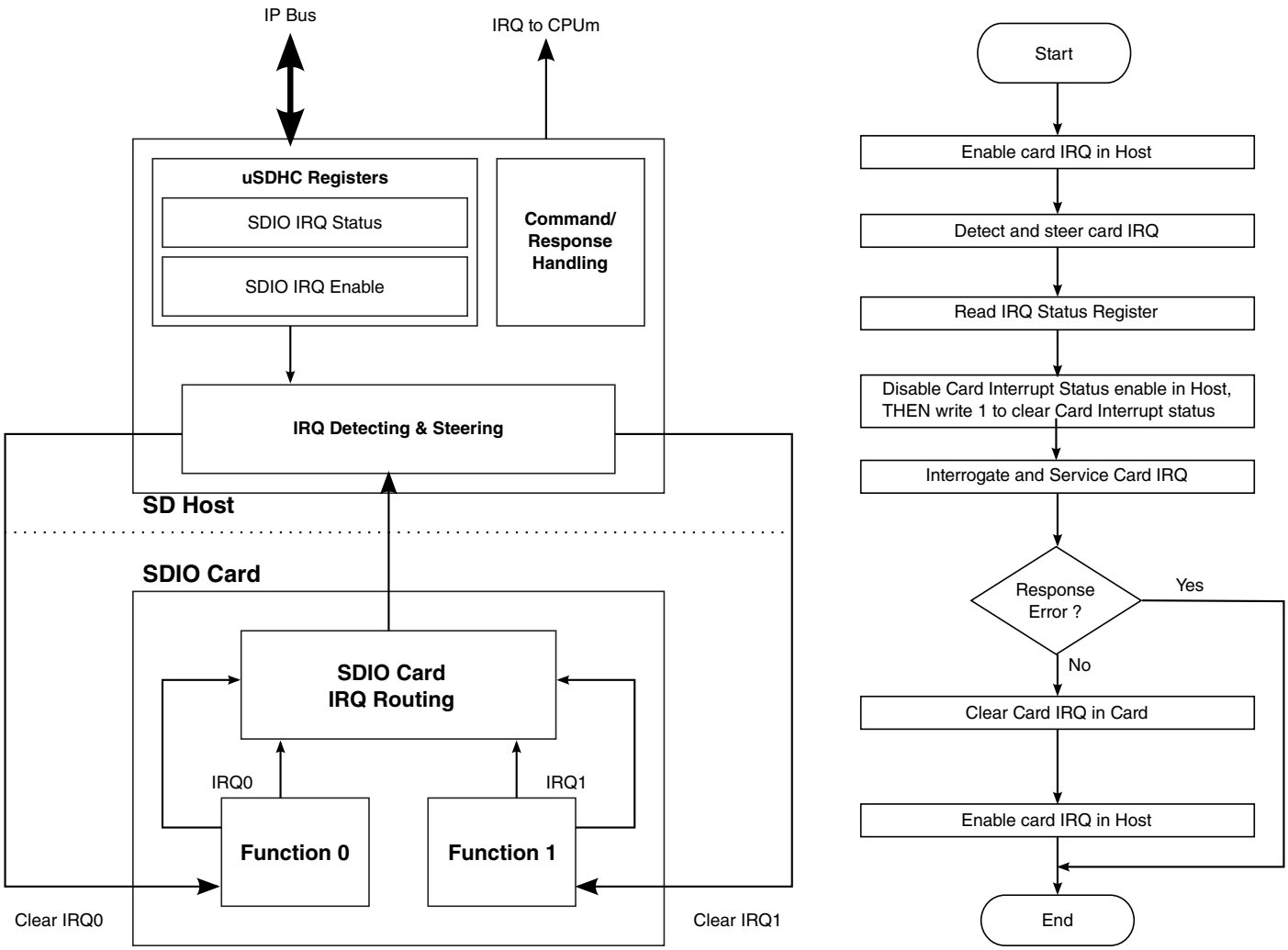


Figure 67-12. Card Interrupt Scheme and Card Interrupt Detection and Handling Procedure

67.4.7 Card Insertion and Removal Detection

The uSDHC uses either the DATA3 pin or the CD_B pin to detect card insertion or removal. When there is no card on the MMC/SD bus, the DATA3 will be pulled to a low voltage level by default.

When any card is inserted to or removed from the socket, the uSDHC detects the logic value changes on the DATA3 pin and generates an interrupt. When the DATA3 pin is not used for card detection (for example, it is implemented in GPIO), the CD_B pin must be connected for card detection. Whether DATA3 is configured for card detection or not, the CD_B pin is always a reference for card detection. Whether the DATA3 pin or the CD_B pin is used to detect card insertion, the uSDHC will send an interrupt (if enabled) to inform the Host system that a card is inserted.

67.4.8 Power Management and Wake Up Events

When there is no operation between the uSDHC and the card through the SD bus, the user can completely disable the ipg_clk and ipg_perclk in the chip level clock control module to save power. When the user needs to use the uSDHC to communicate with the card, it can enable the clock and start the operation.

In some circumstances, when the clocks to the uSDHC are disabled, for instance, when the system is in low power mode, there are some events for which the user needs to enable the clock and handle the event. These events are called wakeup interrupts. The uSDHC can generate these interrupt even when there are no clocks enabled. The three interrupts which can be used as wake up events are:

1. Card Removal Interrupt
2. Card Insertion Interrupt
3. Interrupt from SDIO card

The uSDHC offers a power management feature.

By clearing the clock enabled bits in the System Control Register, the clocks are gated in the low position to the uSDHC. For maximum power saving, the user can disable all the clocks to the uSDHC when there is no operation in progress.

These three wake up events (or wakeup interrupts) can also be used to wake up the system from low-power modes.

NOTE

To make the interrupt a wakeup event, when all the clocks to the uSDHC are disabled or when the whole system is in low power mode, the corresponding wakeup enabled bit needs to be

set. Refer to Protocol Control (uSDHC_PROT_CTRL) for more information on the uSDHC Protocol Control register.

67.4.8.1 Setting Wake Up Events

For the uSDHC to respond to a wakeup event, the software must set the respective wakeup enable bit before the CPU enters sleep mode.

Before the software disables the host clock, it should ensure that all of the following conditions have been met:

- No Read or Write Transfer is active
- Data and Command lines are not active
- No interrupts are pending
- Internal data buffer is empty

67.4.9 MMC fast boot

The Embedded MultiMediaCard (eMMC4.3) specification adds a fast boot feature which requires hardware support.

In boot operation mode, the master (MultiMediaCard host) can read boot data from the slave (MMC device) by keeping CMD line low after power-on, or sending CMD0 with argument + 0xFFFFFFFA (optional for slave), before issuing CMD1.

There are two types of fast boot mode, boot operation and alternative boot operation, in the eMMC4.3 specification. Each type also has with-acknowledge and without-acknowledge modes.

NOTE

For the eMMC4.3 card setting, please see the eMMC4.3 specification.

67.4.9.1 Boot operation

NOTE

For the purposes of this documentation, fast boot is called "normal fast boot mode".

If the CMD line is held LOW for 74 clock cycles and more after power-up before the first command is issued, the slave recognizes that boot mode is being initiated and starts preparing boot data internally.

Within 1 second after the CMD line goes LOW, the slave starts to send the first boot data to the master on the DATA line(s). The master must keep the CMD line LOW to read all of the boot data.

If boot acknowledge is enabled, the slave has to send acknowledge pattern '010' to the master within 50ms after the CMD line goes LOW. If boot acknowledge is disabled, the slave will not send out acknowledge pattern '010'.

The master can terminate boot mode with the CMD line HIGH.

Boot operation will be terminated when all contents of the enabled boot data are sent to the master. After boot operation is executed, the slave shall be ready for CMD1 operation and the master needs to start a normal MMC initialization sequence by sending CMD1.

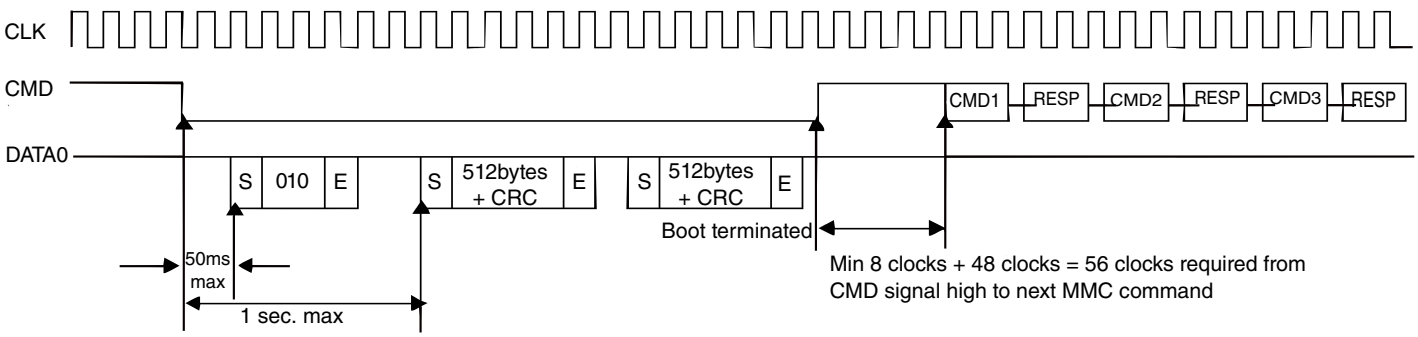


Figure 67-13. MultiMediaCard state diagram (normal boot mode)

67.4.9.2 Alternative boot operation

This boot function is optional for the device. If bit 0 in the extended CSD byte[228] is set to '1', the device supports the alternative boot operation.

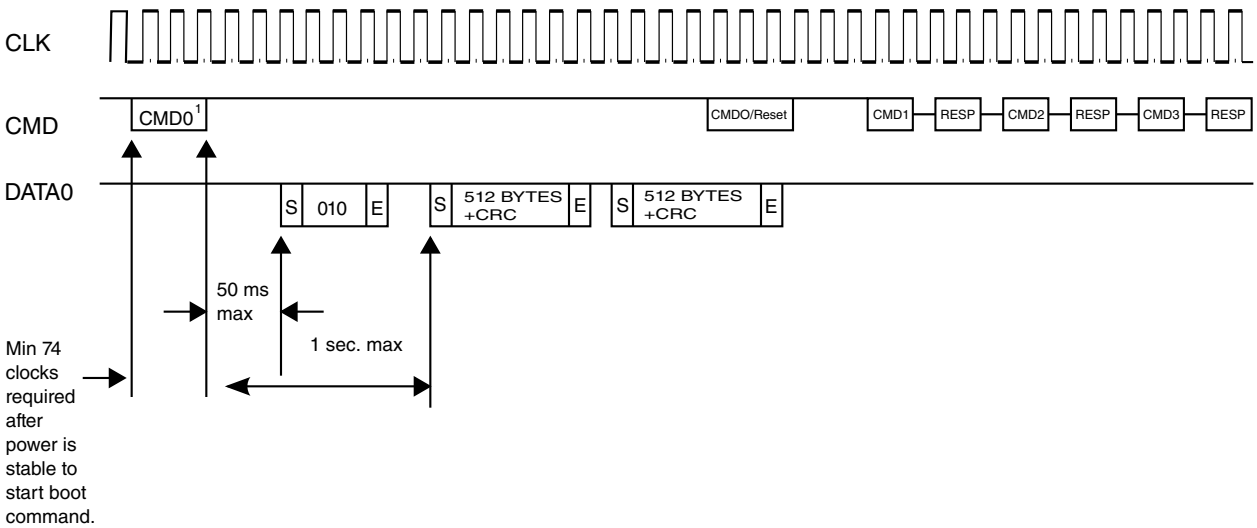
After power-up, if the host issues CMD0 with the argument of 0xFFFFFFFA after 74 clock cycles, before CMD1 is issued or the CMD line goes low, the slave recognizes that boot mode is being initiated and starts preparing boot data internally.

Within 1 second after CMD0 with the argument of 0xFFFFFFFA is issued, the slave starts to send the first boot data to the master on the DATA line(s).

If boot acknowledge is enabled, the slave has to send the acknowledge pattern '010' to the master within 50ms after the CMD0 with the argument of 0xFFFFFFFA is received. If boot acknowledge is disabled, theslave will not send out acknowledge pattern '010'.

The master can terminate boot mode by issuing CMD0 (Reset).

Boot operation will be terminated when all contents of the enabled boot data are sent to the master. After boot operation is executed, the slave shall be ready for CMD1 operation and the master needs to start a normal MMC initialization sequence by sending CMD1.



NOTE 1. CMD0 with argument 0xFFFFFFFA

Figure 67-14. MultiMediaCard state diagram (alternative boot mode)

67.5 Initialization/Application of uSDHC

All communication between the system and cards are controlled by the host. The host sends commands of two types: broadcast and addressed (point-to-point).

Broadcast commands are intended for all cards, such as GO_IDLE_STATE, SEND_OP_COND, ALL_SEND_CID. In Broadcast mode, all cards are in the open-drain mode to avoid bus contention. Refer to Commands for MMC/SD/SDIO for the commands of bc and bcr categories.

After the Broadcast command CMD3 is issued, the cards enter standby mode. Addressed type commands are used from this point. In this mode, the CMD/DATA I/O pads will turn to push-pull mode, to have the driving capability for maximum frequency operation. Refer to Commands for MMC/SD/SDIO for the commands of ac and adtc categories.

67.5.1 Command Send & Response Receive Basic Operation

Assuming the data type WORD is an unsigned 32-bit integer, the below flow is a guideline for sending a command to the card(s):

```
send_command(cmd_index, cmd_arg, other requirements)
{
WORD wCmd; // 32-bit integer to make up the data to write into Transfer Type register, it is
recommended to implement in a bit-field manner
wCmd = (<cmd_index> & 0x3f) >> 24; // set the first 8 bits as '00'+<cmd_index>
set CMDTYP, DPSEL, CICEN, CCCEN, RSTTYP, DTDSEL accorind to the command index;
if (internal DMA is used) wCmd |= 0x1;
if (multi-block transfer) {
          set MSBSEL bit;
          if (finite block number) {
                    set BCEN bit;
                    if (auto12 command is to use) set AC12EN bit;
          }
}
write_reg(CMDARG, <cmd_arg>); // configure the command argument
write_reg(XFERTYP, wCmd); // set Transfer Type register as wCmd value to issue the command
}
wait_for_response(cmd_index)
{
while (CC bit in IRQ Status register is not set); // wait until Command Complete bit is set
read IRQ Status register and check if any error bits about Command are set
if (any error bits are set) report error;
write 1 to clear CC bit and all Command Error bits;
}
```

For the sake of simplicity, the function wait_for_response is implemented here by means of polling. For an effective and formal way, the response is usually checked after the Command Complete Interrupt is received. When doing this, make sure the corresponding interrupt status bits are enabled.

For some scenarios, the response time-out is expected. For instance, after all cards respond to CMD3 and go to the Standby State, no response to the Host when CMD2 is sent. The Host Driver will deal with "fake" errors like this with caution.

67.5.2 Card Identification Mode

When a card is inserted to the socket or the card was reset by the host, the host needs to validate the operation voltage range, identify the cards, request the cards to publish the Relative Card Address (RCA) or to set the RCA for the MMC cards.

67.5.2.1 Card Detect

See the figure below for a flow diagram showing the detection of MMC, SD and SDIO cards using the uSDHC.

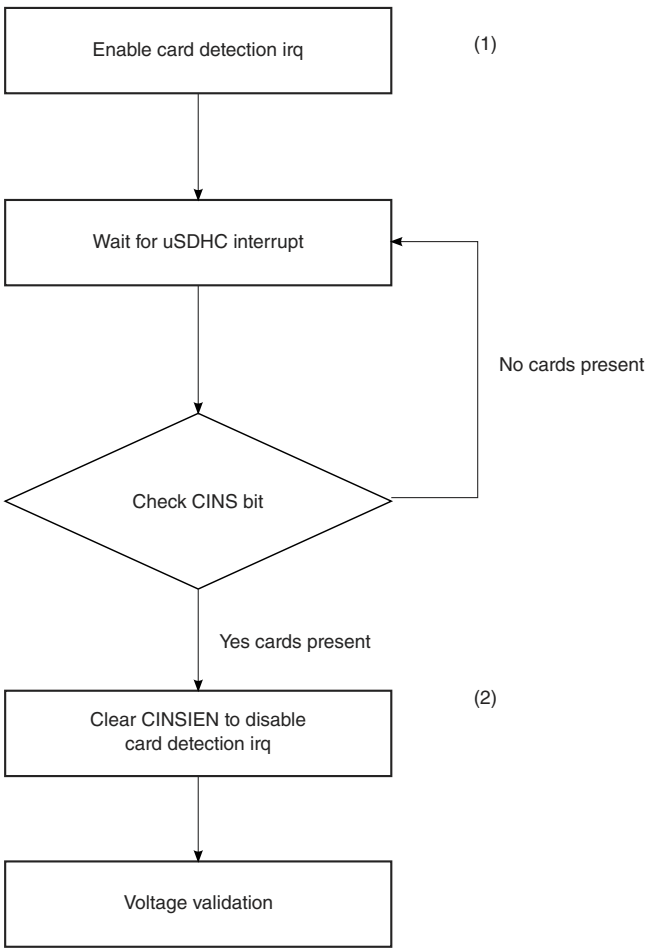


Figure 67-15. Flow Diagram for Card Detection

Here is the card detect sequence:

- Set the CINSIEN bit to enable card detection interrupt
- When an interrupt from the uSDHC is received, check the CINS bit in the Interrupt Status register to see if it was caused by card insertion
- Clear the CINSIEN bit to disable the card detection interrupt and ignore all card insertion interrupts afterwards

67.5.2.2 Reset

The host consists of three types of resets:

- Hardware reset (Card and Host) which is driven by POR (Power On Reset)

- Software reset (Host Only) is initiated by the write operation on the RSTD, RSTC, or RSTA bits of the System Control register to reset the data part, command part, or all parts of the Host Controller, respectively
- Card reset (Card Only). The command, "Go_Idle_State" (CMD0), is the software reset command for all types of MMC cards, SD Memory cards. This command sets each card into the Idle State regardless of the current card state. For an SD I/O Card, CMD52 is used to write an I/O reset in the CCCR. The cards are initialized with a default relative card address (RCA=0x0000) and with a default driver stage register setting (lowest speed, highest driving current capability).

After the card is reset, the host needs to validate the voltage range of the card. See the figure below for the software flow to reset both the uSDHC and the card.

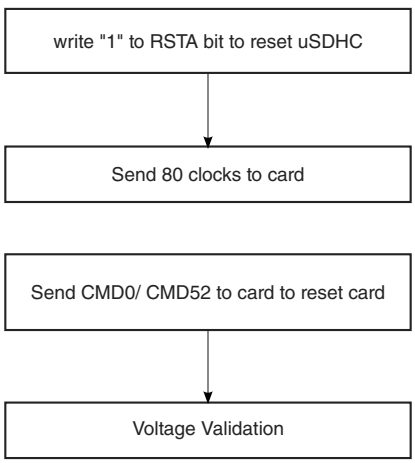


Figure 67-16. Flow Chart for Reset of the uSDHC and SD I/O Card

```
software_reset()
{
set_bit(SYSCTRL, RSTA); // software reset the Host
set DTOCV and SDCLKFS bit fields to get the CLK of frequency around 400kHz
configure IO pad to set the power voltage of external card to around 3.0V
poll bits CIHB and CDIHB bits of PRSSTAT to wait both bits are cleared
set_bit(SYSCTRL, INTIA); // send 80 clock ticks for card to power up
send_command(CMD_GO_IDLE_STATE, <other parameters>); // reset the card with CMD0
or send_command(CMD_IO_RW_DIRECT, <other parameters>);
}
```

67.5.2.3 Voltage Validation

All cards should be able to establish communication with the host using any operation voltage in the maximum allowed voltage range specified in the card specification. However, the supported minimum and maximum values for Vdd are defined in the Operation Conditions Register (OCR) and may not cover the whole range.

Cards that store the CID and CSD data in the preload memory are only able to communicate this information under data transfer Vdd conditions. This means if the host and card have non-common Vdd ranges, the card will not be able to complete the identification cycle, nor will it be able to send CSD data.

Therefore, a special command Send_Op_Cont (CMD1 for MMC), SD_Send_Op_Cont (ACMD41 for SD Memory) and IO_Send_Op_Cont (CMD5 for SD I/O) is used. The voltage validation procedure is designed to provide a mechanism to identify and reject cards which do not match the Vdd range(s) desired by the host. This is accomplished by the host sending the desired Vdd voltage window as the operand of this command. Cards that can't perform the data transfer in the specified range must discard themselves from further bus operations and go into the Inactive State. By omitting the voltage range in the command, the host can query each card and determine the common voltage range before sending out-of-range cards into the Inactive State. This query should be used if the host is able to select a common voltage range or if a notification shall be sent to the system when a non-usable card in the stack is detected.

The following steps show how to perform voltage validation when a card is inserted:

```
voltage_validation(voltage_range_arguement)
{
label the card as UNKNOWN;
send_command(IO_SEND_OP_COND, 0x0, <other parameters are omitted>); // CMD5, check SDIO
operation voltage, command argument is zero
if (RESP_TIMEOUT != wait_for_response(IO_SEND_OP_COND)) { // SDIO command is accepted
          if (0 < number of IO functions) {
                    label the card as SDIO;
                    IORDY = 0;
                    while (!(IORDY in IO OCR response)) { // set voltage range for each IO
function
                              send_command(IO_SEND_OP_COND, <voltage range>, <other
parameter>);
                              wait_for_response(IO_SEND_OP_COND);
                    } // end of while ...
          } // end of if (0 < ...
          if (memory part is present inside SDIO card) Label the card as SDCombo; // this is
an
SD-Combo card
} // end of if (RESP_TIMEOUT ...
if (the card is labelled as SDIO card) return; // card type is identified and voltage range
is
set, so exit the function;
send_command(APP_CMD, 0x0, <other parameters are omitted>); // CMD55, Application specific
CMD
prefix
if (no error calling wait_for_response(APP_CMD, <...>) { // CMD55 is accepted
          send_command(SD_APP_OP_COND, <voltage range>, <...>); // ACMD41, to set voltage
range
for memory part or SD card
          wait_for_response(SD_APP_OP_COND); // voltage range is set
          if (card type is UNKNOWN) label the card as SD;
          return; //
} // end of if (no error ...
else if (errors other than time-out occur) { // command/response pair is corrupted
          deal with it by program specific manner;
} // of else if (response time-out
else { // CMD55 is refuse, it must be MMC card  if (card is already labelled as SDCombo)
{ //
change label
```

```
                    re-label the card as SDIO;
                    ignore the error or report it;
                    return; // card is identified as SDIO card
          } // of if (card is ...
          send_command(SEND_OP_COND, <voltage range>, <...>);
          if (RESP_TIMEOUT == wait_for_response(SEND_OP_COND)) { // CMD1 is not accepted,
either
                    label the card as UNKNOWN;
                    return;
          } // of if (RESP_TIMEOUT ...
} // of else
}
```

67.5.2.4 Card Registry

Card registry for the MMC and SD/SDIO/SD Combo cards are different. For the SD Card, the Identification process starts at a clock rate lower than 400 kHz and the power voltage higher than 2.7 V (as defined by the Card spec). At this time, the CMD line output drives are push-pull drivers instead of open-drain. After the bus is activated, the host will request the card to send their valid operation conditions.

The response to ACMD41 is the operation condition register of the card. The same command shall be send to all of the new cards in the system. Incompatible cards are put into the Inactive State. The host then issues the command, All_Send_CID (CMD2), to each card to get its unique card identification (CID) number. Cards that are currently unidentified (in the Ready State), send their CID number as the response. After the CID is sent by the card, the card goes into the Identification State.

The host then issues Send_Relative_Addr (CMD3), requesting the card to publish a new relative card address (RCA) that is shorter than the CID. This RCA will be used to address the card for future data transfer operations. Once the RCA is received, the card changes its state to the Standby State. At this point, if the host wants the card to have an alternative RCA number, it may ask the card to publish a new number by sending another Send_Relative_Addr command to the card. The last published RCA is the actual RCA of the card.

The host repeats the identification process with CMD2 and CMD3 for each card in the system until the last CMD2 gets no response from any of the cards in system.

For MMC operation, the host starts the card identification process in open-drain mode with the identification clock rate lower than 400 kHz and the power voltage higher than 2.7 V. The open drain driver stages on the CMD line allow parallel card operation during card identification. After the bus is activated the host will request the cards to send their valid operation conditions (CMD1). The response to CMD1 is the "wired OR" operation on the condition restrictions of all cards in the system. Incompatible cards are sent into the Inactive State. The host then issues the broadcast command All_Send_CID (CMD2), asking all cards for their unique card identification (CID) number. All unidentified cards

(the cards in Ready State) simultaneously start sending their CID numbers serially, while bit-wise monitoring their outgoing bit stream. Those cards, whose outgoing CID bits do not match the corresponding bits on the command line in any one of the bit periods, stop sending their CID immediately and must wait for the next identification cycle. Since the CID is unique for each card, only one card can be successfully send its full CID to the host. This card then goes into the Identification State. Thereafter, the host issues Set_Relative_Addr (CMD3) to assign to the card a relative card address (RCA). Once the RCA is received the card state changes to the Stand-by State, and the card does not react in further identification cycles, and its output driver switches from open-drain to push-pull. The host repeats the process, mainly CMD2 and CMD3, until the host receives a time-out condition to recognize the completion of the identification process.

For operation as MMC cards:

```
card_registry()
{
do { // decide RCA for each card until response time-out
          if(card is labelled as SDCombo or SDIO) { // for SDIO card like device
                    send_command(SET_RELATIVE_ADDR, 0x00, <...>); // ask SDIO card to
publish its
RCA
                    retrieve RCA from response;
          } // end if (card is labelled as SDCombo ...
          else if (card is labelled as SD) { // for SD card
                    send_command(ALL_SEND_CID, <...>);
                    if (RESP_TIMEOUT == wait_for_response(ALL_SEND_CID)) break;
                    send_command(SET_RELATIVE_ADDR, <...>);
                    retrieve RCA from response;
          } // else if (card is labelled as SD ...
          else if (card is labelled as MMC) {
                    send_command(ALL_SEND_CID, <...>);
                    rca = 0x1; // arbitrarily set RCA, 1 here for example
                    send_command(SET_RELATIVE_ADDR, 0x1 << 16, <...>); // send RCA at upper
16
bits
          } // end of else if (card is labelled as MMC ...
} while (response is not time-out);
}
```

67.5.3 Card Access

67.5.3.1 Block Write

67.5.3.1.1 Normal Write

During a block write (CMD24 - 27, CMD60, CMD61), one or more blocks of data are transferred from the host to the card with a CRC appended to the end of each block by the host. If the CRC fails, the card shall indicate the failure on the DATA line. The transferred data will be discarded and not written, and all further transmitted blocks (in multiple block write mode) will be ignored.

If the host uses partial blocks whose accumulated length is not block aligned and block misalignment is not allowed (CSD parameter WRITE_BLK_MISALIGN is not set), the card detects the block misalignment error and aborts the programming before the beginning of the first misaligned block. The card sets the ADDRESS_ERROR error bit in the status register, and while ignoring all further data transfer, waits in the Receive-data-State for a stop command. The write operation is also aborted if the host tries to write over a write protected area.

For MMC and SD cards, programming of the CID and CSD registers does not require a previous block length setting. The transferred data is also CRC protected. If a part of the CSD or CID register is stored in ROM, then this unchangeable part must match the corresponding part of the receive buffer. If this match fails, then the card will report an error and not change any register contents.

For all types of cards, some may require long and unpredictable periods of time to write a block of data. After receiving a block of data and completing the CRC check, the card will begin writing and hold the DATA line low if its write buffer is full and unable to accept new data from a new WRITE_BLOCK command. The host may poll the status of the card with a SEND_STATUS command (CMD13) or other means for SDIO cards at any time, and the card will respond with its status. The responded status indicates whether the card can accept new data or whether the write process is still in progress. The host may deselect the card by issuing a CMD7 (to select a different card) to place the card into the Standby State and release the DATA line without interrupting the write operation. When re-selecting the card, it will reactivate the busy indication by pulling DATA to low if the programming is still in progress and the write buffer is unavailable.

The software flow to write to a card incorporates the internal DMA and the write operation is a multi-block write with the Auto CMD12 enabled. For the other two methods (by means of external DMA or CPU polling status) with different transfer methods, the internal DMA parts should be removed and the alternative steps should be straightforward.

The software flow to write to a card is described below:

1. Check the card status, wait until the card is ready for data.
2. Set the card block length/size:
   - For SD/MMC cards, use SET_BLOCKLEN (CMD16)
   - For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT (CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7)
3. Set the uSDHC block length register to be the same as the block length set for the card in Step 2.
4. Set the uSDHC number block register (NOB), nob is 5 (for instance).

5. Disable the buffer write ready interrupt, configure the DMA settings and enable the uSDHC DMA when sending the command with data transfer. The AC12EN bit should also be set.
6. Wait for the Transfer Complete interrupt.
7. Check the status bit to see if a write CRC error occurred, or some another error, that occurred during the auto12 command sending and response receiving.

67.5.3.1.2 DDR Write

uSDHC supports dual data rate mode.

The software flow to write to a card in ddr mode is described as below:

1. Check the card status, wait until the card is ready for data.
2. For eMMC4.4 card, block length only can be set to 512byte.
3. Set the uSDHC number block register (NOB), nob is 5 (for instance).
4. Set eMMC4.4 card to high speed mode, use SWITCH(CMD6).
5. Set eMMC4.4 card bus with (4-bit /8-bit ddr mode), use SWITCH(CMD6).
6. Disable the buffer write ready interrupt, configure the DMA settings and enable the uSDHC DMA when sending the command with data transfer.The DDR_EN bit should be set. The AC12EN bit should also be set.
7. Wait for the Transfer Complete interrupt.
8. Check the status bit to see if a write CRC error occurred, or some another error, that occurred during the auto12 command sending and response receiving.

67.5.3.1.3 Write with Pause

The write operation can be paused during the transfer. Instead of stopping the CLK at any time to pause all the operations, which is also inaccessible to the Host Driver, the Driver can set the Stop At Block Gap Request(SABGREQ) bit in the Protocol Control register to pause the transfer between the data blocks. As there is no time-out condition in a write operation during the data blocks, a write to all types of cards can be paused in this way, and if the DATA0 line is not required to de-assert to release the busy state, no suspend command is needed.

Like in the flow described in Normal Write, the write with pause is shown with the same kind of write operation:

1. Check the card status, wait until card is ready for data.
2. Set the card block length/size:

- For SD/MMC, use SET_BLOCKLEN (CMD16)
- For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT(CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7)

3. Set the uSDHC block length register to be the same as the block length set for the card in Step 2.
4. Set the uSDHC number block register (NOB), nob is 5 (for instance).
5. Disable the buffer write ready interrupt, configure the DMA settings and enable the uSDHC DMA when sending the command with data transfer. The AC12EN bit should also be set.
6. Set the SABGREQ bit.
7. Wait for the Transfer Complete interrupt.
8. Clear the SABGREQ bit.
9. Check the status bit to see if a write CRC error occurred.
10. Set the CREQ bit to continue the write operation.
11. Wait for the Transfer Complete interrupt.
12. Check the status bit to see if a write CRC error occurred, or some another error, that occurred during the auto12 command sending and response receiving.

The number of blocks left during the data transfer is accessible by reading the contents of the BLKCNT field in the Block Attribute register. As the data transfer and the setting of the SABGREQ bit are concurrent, and the delay of register read and the register setting, the actual number of blocks left may not be exactly the value read earlier. The Driver shall read the value of BLKCNT after the transfer is paused and the Transfer Complete interrupt is received.

It is also possible the last block has begun when the Stop At Block Gap Request is sent to the buffer. In this case, the next block gap is actually the end of the transfer. These types of requests are ignored and the Driver should treat this as a non-pause transfer and deal with it as a common write operation.

When the write operation is paused, the data transfer inside the Host System is not stopped, and the transfer is active until the data buffer is full. Because of this (if not needed), it is recommended to avoid using the Suspend Command for the SDIO card. This is because when such a command is sent, the uSDHC thinks the System will switch to another function on the SDIO card, and flush the data buffer. The uSDHC takes the Resume Command as a normal command with data transfer, and it is left for the Driver to set all the relevant registers before the transfer is resumed. If there is only one block to send when the transfer is resumed, the MSBSEL and BCEN bits of the Transfer Type register are set as well as the AC12EN bit. However, the uSDHC will automatically send a CMD12 to mark the end of the multi-block transfer.

67.5.3.2 Block Read

67.5.3.2.1 Normal Read

For block reads, the basic unit of data transfer is a block whose maximum size is stored in areas defined by the corresponding card specification. A CRC is appended to the end of each block, ensuring data transfer integrity. The CMD17, CMD18, CMD53, CMD60, CMD61, and so on, can initiate a block read. After completing the transfer, the card returns to the Transfer State. For multi blocks read, data blocks will be continuously transferred until a stop command is issued.

The software flow to read from a card incorporates the internal DMA and the read operation is a multi-block read with the Auto CMD12 enabled. For the other two methods (by means of external DMA or CPU polling status) with different transfer methods, the internal DMA parts should be removed and the alternative steps should be straightforward.

The software flow to read from a card is described below:

1. Check the card status, wait until card is ready for data.
2. Set the card block length/size:
   - For SD/MMC, use SET_BLOCKLEN (CMD16)
   - For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT(CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7)
3. Set the uSDHC block length register to be the same as the block length set for the card in Step 2.
4. Set the uSDHC number block register (NOB), nob is 5 (for instance).
5. Disable the buffer read ready interrupt, configure the DMA settings and enable the uSDHC DMA when sending the command with data transfer. The AC12EN bit should also be set.
6. Wait for the Transfer Complete interrupt.
7. Check the status bit to see if a read CRC error occurred, or some another error, occurred during the auto12 command sending and response receiving.

67.5.3.2.2 DDR Read

uSDHC supports dual data rate mode.

The software flow to write to a card in ddr mode is described below:

1. Check the card status, wait until the card is ready for data.

2. For eMMC4.4 card, block length only can be set to 512byte.
3. Set the uSDHC number block register (NOB), nob is 5 (for instance).
4. Set eMMC4.4 card to high speed mode, use SWITCH(CMD6).
5. Set eMMC4.4 card bus with (4-bit /8-bit ddr mode), use SWITCH(CMD6).
6. Disable the buffer write ready interrupt, configure the DMA settings and enable the uSDHC DMA when sending the command with data transfer.The DDR_EN bit should be set. The AC12EN bit should also be set.
7. Wait for the Transfer Complete interrupt.
8. Check the status bit to see if a write CRC error occurred, or some another error, that occurred during the auto12 command sending and response receiving.

67.5.3.2.3 Read with Pause

The read operation is not generally able to pause. Only the SDIO card (and SDCombo card working under I/O mode) supporting the Read Wait feature can pause during the read operation. If the SDIO card supports Read Wait (SRW bit in CCCR register is 1), the Driver can set the SABGREQ bit in the Protocol Control register to pause the transfer between the data blocks.

Before setting the SABGREQ bit, make sure the RWCTL bit in the Protocol Control register is set, otherwise the uSDHC will not assert the Read Wait signal during the block gap and data corruption occurs. It is recommended to set the RWCTL bit once the Read Wait capability of the SDIO card is recognized.

Like in the flow described in Normal Read, the read with pause is shown with the same kind of read operation:

1. Check the SRW bit in the CCR register on the SDIO card to confirm the card supports Read Wait.
2. Set the RWCTL bit.
3. Check the card status and wait until the card is ready for data.
4. Set the card block length/size:
   - For SD/MMC, use SET_BLOCKLEN (CMD16)
   - For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT(CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7)
5. Set the uSDHC block length register to be the same as the block length set for the card in Step 2.
6. Set the uSDHC number block register (NOB), nob is 5 (for instance).
7. Disable the buffer read ready interrupt, configure the DMA setting and enable the uSDHC DMA when sending the command with data transfer. The AC12EN bit should also be set
8. Set the SABGREQ bit.

9. Wait for the Transfer Complete interrupt.
10. Clear the SABGREQ bit.
11. Check the status bit to see if read CRC error occurred.
12. Set the CREQ bit to continue the read operation.
13. Wait for the Transfer Complete interrupt.
14. Check the status bit to see if a read CRC error occurred, or some another error, occurred during the auto12 command sending and response receiving.

Like the write operation, it is possible to meet the ending block of the transfer when paused. In this case, the uSDHC will ignore the Stop At Block Gap Request and treat it as a command read operation.

Unlike the write operation, there is no remaining data inside the buffer when the transfer is paused. All data received before the pause will be transferred to the Host System. No matter if the Suspend Command is sent or not, the internal data buffer is not flushed.

If the Suspend Command is sent and the transfer is later resumed by means of a Resume Command, the uSDHC takes the command as a normal one accompanied with data transfer. It is left for the Driver to set all the relevant registers before the transfer is resumed. If there is only one block to send when the transfer is resumed, the MSBSEL and BCEN bits of the Transfer Type register are set, as well as the AC12EN bit. However, the uSDHC will automatically send the CMD12 to mark the end of multi-block transfer.

67.5.3.2.4 DLL (Delay Line) in Read Path

The DLL(Delay Line) is newly added to assist in sampling read data. The DLL provides the ability to programmatically select a quantized delay (in fractions of the clock period) regardless of on-chip variations such as process, voltage and temperature (PVT).

The reasons why the DLL is needed for uSDHC are 1.) the path of read data traveling from card to host varies. 2.) in SD/MMC DDR mode the minimum input setup and hold time are both at 2.5 ns. The data sampling window is so small that the delay of loopback clock needs to be accurate and consistent regardless of PVT. The DLL takes the divided card_clk as the reference clock and loopback clock as the input clock. It then generates a delayed version of the input clock according to the programmed target delay.

The DLL can be disabled or bypassed, and it can also be manually set for a fixed delay in override mode. The override value set is the number of delay cells. In override mode, there is no need to set the DLL_enable. Another DLL mode is target value mode. In this mode, the DLL will automatically adjust the number of delay cells according to target value set by user and PVT changes. Be aware that target value is in units of 1/32 of the

clock reference period. If the card_clk is 100Mhz, then the reference clock period is 10ns, setting target vaule of 16 means 5ns =(16/32)*10ns. Software can disable automatic update by setting dll_gate_update bit. Please refer to Figure 67-17.

Since the user may change the frequency of the card_clk from time to time by changing SDCLKFS[7:0]/DVS[3:0], the software must adjust the delay value to ensure it works correctly when the reference clock (card_clk) is changed. The following is the correct flow, which should be ignored if DLL_CTRL_ENABLE is not set.

Step 1: Set DLL_CTRL_RESET bit

Step 2: Configure the SDCLKFS[7:0] and DVS[3:0]

Step 3: Wait until SDSTB is asserted

Step 4: clear DLL_CTRL_RESET bit

Step 5: Wait until both DLL_STS_SLV_LOCK and DLL_STS_REF_LOCK are asserted

Step 6: set DLL_CTRL_SLV_FORCE_UPD

Step 7: clear DLL_CTRL_SLV_FORCE_UPD

NOTE:Software should make sure the DLL_CTRL_SLV_FORCE_UPD is lasted for at least one card_clk. So software may need to add some delay between step6 and step7.

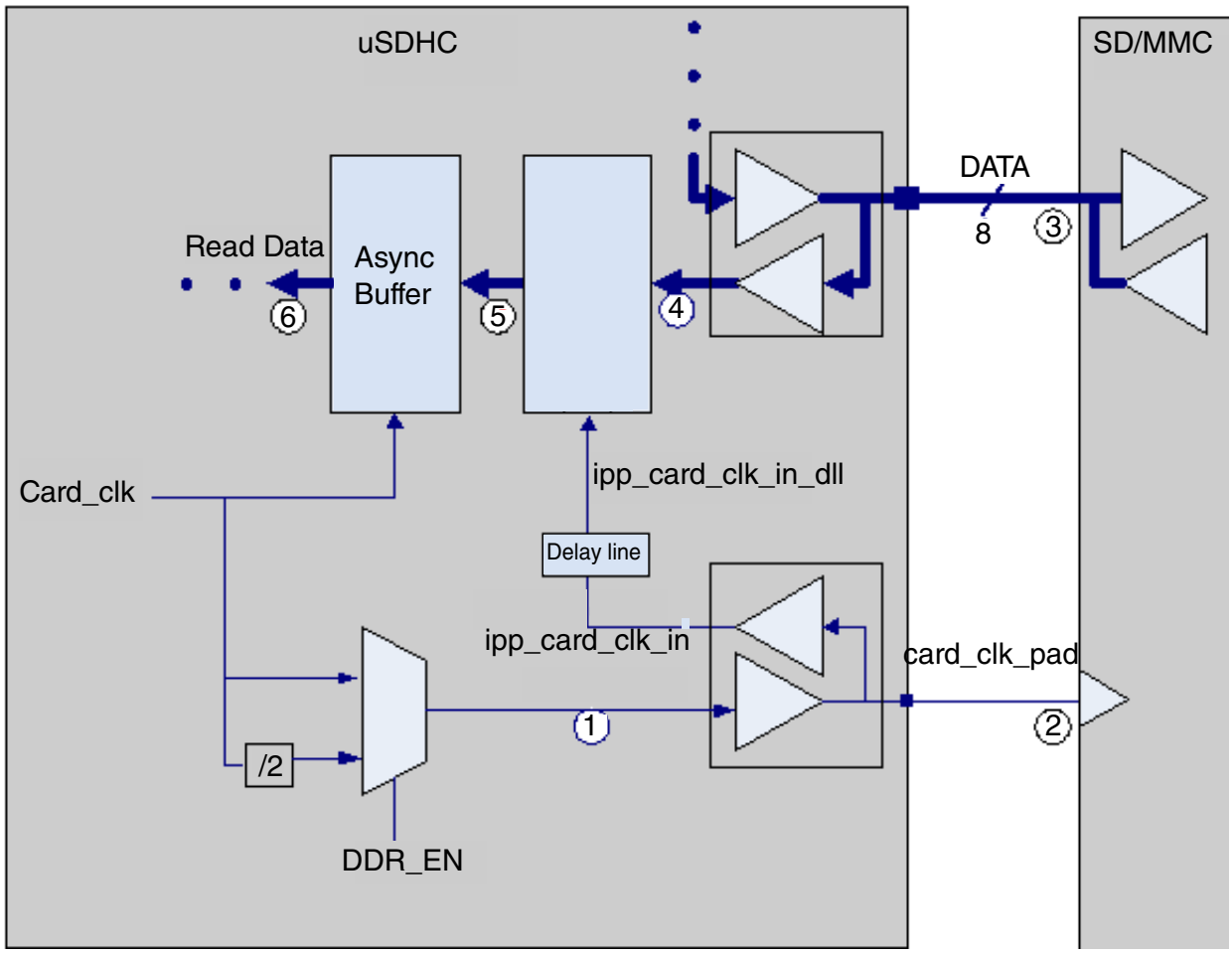


Figure 67-17. DLL(Delay Line) in Read Path

67.5.3.3 Suspend Resume

The uSDHC supports the Suspend Resume operations of SDIO cards, although slightly differently than the suggested implementation of Suspend in the SDIO card specification.

67.5.3.3.1 Suspend

After setting the SABGREQ bit, the Host Driver may send a Suspend command to switch to another function of the SDIO card. The uSDHC does not monitor the content of the response, so it doesn't know if the Suspend command succeeded or not. Accordingly, it doesn't de-assert Read Wait for read pause. To solve this problem, the Driver shall not mark the Suspend command as a "Suspend", (i.e. setting the CMDTYP bits to 01). Instead, the Driver shall send this command as if it were a normal command, and only

when the command succeeds, and the BS bit is set in the response, can the Driver send another command marked as "Suspend" to inform the uSDHC that the current transfer is suspended. As shown in the following sequence for Suspend operation:

1. Set the SABREQ bit to pause the current data transfer at block gap.
2. After the BGE bit is set, send the Suspend command to suspend the active function. The CMDTYP bit field must be 2'b00.
3. Check the BS bit of the CCCR in the response. If it is 1, repeat this step until the BS bit is cleared or abandon the suspend operation according to the Driver strategy.
4. Send another normal I/O command to the suspended function. The CMDTYP of this command must be 2'b01, so the uSDHC can detect this special setting and be informed that the paused operation has successfully suspended. If the paused transfer is a read operation, the uSDHC stops driving DATA2 and goes to the idle state.
5. Save the context registers in the system memory for later use, including the DMA System Address Register (for internal DMA operation), and the Block Attribute Register.
6. Begin operation for another function on the SDIO card.

67.5.3.3.2 Resume

To resume the data transfer, a Resume command shall be issued:

1. To resume the suspended function, restore the context register with the saved value in step #5 of the Suspend operation above.
2. Send the Resume command. In the Transfer Type register, all bit fields are set to the value as if this were another ordinary data transfer, instead of a transfer resume (except the CMDTYP is set to 2'b10).
3. If the Resume command has responded, the data transfer will be resumed.

67.5.3.4 ADMA Usage

To use the ADMA in a data transfer, the Host Driver must prepare the correct descriptor chain prior to sending the read/write command.

The steps to prepare the correct descriptor chain are:

1. Create a descriptor to set the data length that the current descriptor group is about to transfer. The data length should be even numbers of the block size.
2. Create another descriptor to transfer the data from the address setting in this descriptor. The data address must be at a page boundary (4kB address aligned).
3. If necessary, create a Link descriptor containing the address of the next descriptor. The descriptor group is created in steps 1 ~ 3.

4. Repeat steps 1 ~ 3 until all descriptors are created.
5. In the last descriptor, set the End flag to 1 and make sure the total length of all descriptors match the product of the block size and block number configured in the Block Attribute Register.
6. Set the ADMA System Address Register to the address of the first descriptor and set the DMAS field in the Protocol Control Register to 01 to select the ADMA.
7. Issue a write or read command with the DMAEN bit set to 1 in the Transfer Type Register.

Steps 1 ~ 5 are independent of step 6, so step 6 can finish before steps 1 ~ 5. Regarding the descriptor configuration, it is recommended not to use the Link descriptor as it requires extra system memory access.

67.5.3.5 Transfer Error

67.5.3.5.1 CRC Error

It is possible at the end of a block transfer, that a write CRC status error or read CRC error occurs. For this type of error the latest block received shall be discarded. This is because the integrity of the data block is not guaranteed. It is recommended to discard the following data blocks and re-transfer the block from the corrupted one.

For a multi-block transfer, the Host Driver shall issue a CMD12 to abort the current process and start the transfer by a new data command. In this scenario, even when the AC12EN and BCEND bits are set, the uSDHC does not automatically send a CMD12 because the last block is not transferred. On the other hand, if it is within the last block that the CRC error occurs, an Auto CMD12 will be sent by the uSDHC. In this case, the Driver shall re-send or re-obtain the last block with a single block transfer.

67.5.3.5.2 Internal DMA Error

During the data transfer with internal Simple DMA, if the DMA engine encounters some error on the AHB bus, the DMA operation is aborted and DMA Error interrupt is sent to the Host System. When acknowledged by such an interrupt, the Driver shall calculate the start address of data block in which the error occurs.

The start address can be calculated by either:

1. Reading the DMA System Address register. The error occurs during the previous burst. Taking the block size, the previous burst length and the start address of the next burst transfer into account, it is straight forward to obtain the start address of the corrupted block.

2. Reading the BLKCNT field of the Block Attribute register. By the number of blocks left, the total number to transfer, the start address of transfer, and the size of each block, the start address of corrupted block can be determined. When the BCEN bit is not set, the contents of the Block Attribute register does not change, so this method does not work.

When a DMA error occurs, it is recommended to abort the current transfer by means of a CMD12 (for multi block transfer), apply a reset for data, and re-start the transfer from the corrupted block to recover from the error.

67.5.3.5.3 Transfer ADMA Error

There are 3 kinds of possible ADMA errors. The AHB transfer, invalid descriptor, and data-length mismatch errors. Whenever these errors occur, the DMA transfer stops and the corresponding error status bit is set.

For acknowledging the status, the Host Driver should recover the error as shown below and re-transfer from the place of interruption.

1. AHB transfer error: Such errors may occur during data transfer or descriptor fetch. For either scenario, it is recommended to retrieve the transfer context, reset for the data part and re-transfer the block that was corrupted, or the next block if no block is corrupted.
2. Invalid descriptor error: For such errors, it is recommended to retrieve the transfer context, reset for the data part and re-create the descriptor chain from the invalid descriptor and issue a new transfer. As the data to transfer now may be less than the previous setting, the data length configured in the new descriptor chain should match the new value.
3. Data-length mismatch error: It is similar to recover from this error. The Host Driver polls relating registers to retrieve the transfer context, apply a reset for the data part, configure a new descriptor chain, and make another transfer if there is data left. Like the previous scenario of the invalid descriptor error, the data length must match the new transfer.

67.5.3.5.4 Auto CMD12 Error

After the last block of the multi block transfer is sent or received, and the AC12EN bit is set when the data transfer is initiated by the data command, the uSDHC automatically sends a CMD12 to the card to stop the transfer.

When errors with this command occur, it is recommended to the Driver to deal with the situations in the following manner:

1. Auto CMD12 response time-out. It is not certain whether the command is accepted by the card or not. The Driver should clear the Auto CMD12 error status bits and re-send the CMD12 until it is accepted by the card.
2. Auto CMD12 response CRC error. Since card responds to the CMD12, the card will abort the transfer. The Driver may ignore the error and clear the error status bit.
3. Auto CMD12 conflict error or not sent. The command is not sent, so the Driver shall send a CMD12 manually.

67.5.3.6 Card Interrupt

The external cards can inform the Host Controller by means of some special signals. For the SDIO card, it can be the low level on the DATA1 line during some special period. The uSDHC only monitors the DATA1 line and supports the SDIO interrupt.

When the SDIO interrupt is captured by the uSDHC, and the Host System is informed by the uSDHC asserting the uSDHC interrupt line, the interrupt service from the Host Driver is called.

As the interrupt source is controlled by the external card, the interrupt from the SDIO card must be serviced before the CINT bit is cleared by written 1. Refer to Card Interrupt Handling for the card interrupt handling flow.

67.5.4 Switch Function

SD/MMC cards can transfer data at bus widths other than 1-bit. Different speed mode are also defined. To enable these features, a "switch" command shall be issued by the Host Driver.

For SDIO cards, the high speed mode/DDR50/SDR50/SDR104 are enabled by writing the EHS bit in the CCCR register after the SHS bit is confirmed. For SD cards, the high speed mode/DDR50/SDR50/SDR104 are queried and enabled by a CMD6 (with the mnemonic symbol as SWITCH_FUNC). For MMC cards , the high speed mode/HS200 are queried by a CMD8 and enabled by a CMD6 (with the mnemonic symbol as SWITCH).

The SDR4-bit, SDR8-bit ,DDR4-bit and DDR8-bit width of the MMC is also enabled by the SWITCH command, but with a different argument.

These new functions can also be disabled by a software reset. For SDIO cards it can be done by setting the RES bit in the CCCR register. For other cards, it can be accomplished by issuing a CMD0. This method of restoring to the normal mode is not recommended because a complete identification process is needed before the card is ready for data transfer.

For the sake of simplicity, the following pseudocode examples do not show current capability check, which is recommended in the function switch process.

67.5.4.1 Query, Enable and Disable SDIO High Speed Mode

```
enable_sdio_high_speed_mode(void)
{
send CMD52 to query bit SHS at address 0x13;
if (SHS bit is '0') report the SDIO card does not support high speed mode and return;
send CMD52 to set bit EHS at address 0x13 and read after write to confirm EHS bit is set;
change clock divisor value or configure the system clock feeding into uSDHC to generate the
card_clk of around 50MHz;
(data transactions like normal peers)
}
disable_sdio_high_speed_mode(void)
{
send CMD52 to clear bit EHS at address 0x13 and read after write to confirm EHS bit is
cleared;
change clock divisor value or configure the system clock feeding into uSDHC to generate the
card_clk of the desired value below 25MHz;
(data transactions like normal peers)
}
```

67.5.4.2 Query, Enable and Disable SD High Speed Mode/SDR50/ SDR104/DDR50

```
enable_sd_speed_mode(void)
{
set BLKCNT field to 1 (block), set BLKSIZE field to 64 (bytes);
send CMD6, with argument 0xFFFFFx and read 64 bytes of data accompanying the R1 response;
(high speed mode,x=1; SDR50,x=2; SDR104,x=3; DDR50,x=4;)
wait data transfer done bit is set;
check if the bit x of received 512 bits is set;
if (bit 401 is '0') report the SD card does not support high speed mode and return;
if (bit 402 is '0') report the SD card does not support SDR50 mode and return;
if (bit 403 is '0') report the SD card does not support SDR104 mode and return;
if (bit 404 is '0') report the SD card does not support DDR50 mode and return;
send CMD6, with argument 0x80FFFFFx and read 64 bytes of data accompanying the R1 response;
(high speed mode,x=1; SDR50,x=2; SDR104 x=3; DDR50 x=4;)
check if the bit field 379~376 is 0xF;
if (the bit field is 0xF) report the function switch failed and return;
change clock divisor value or configure the system clock feeding into uSDHC to generate the
card_clk of around 50MHz for high speed mode, 100Mhz for SDR50, 200Mhz for SDR104, 50Mhz for
DDR50;
(data transactions like normal peers)
}
disable_sd_speed_mode(void)
{
set BLKCNT field to 1 (block), set BLKSIZE field to 64 (bytes);
send CMD6, with argument 0x80FFFFF0 and read 64 bytes of data accompanying the R1 response;
check if the bit field 379~376 is 0xF;
```

```
if (the bit field is 0xF) report the function switch failed and return;
change clock divisor value or configure the system clock feeding into uSDHC to generate the
card_clk of the desired value below 25MHz;
(data transactions like normal peers)
}
```

67.5.4.3 Query, Enable and Disable MMC High Speed Mode

```
enable_mmc_high_speed_mode(void)
{
send CMD9 to get CSD value of MMC;
check if the value of SPEC_VER field is 4 or above;
if (SPEC_VER value is less than 4) report the MMC does not support high speed mode and
return;
set BLKCNT field to 1 (block), set BLKSIZE field to 512 (bytes);
send CMD8 to get EXT_CSD value of MMC;
extract the value of CARD_TYPE field to check the 'high speed mode' in this MMC is 26MHz or
52MHz;
send CMD6 with argument 0x1B90100;
send CMD13 to wait card ready (busy line released);
send CMD8 to get EXT_CSD value of MMC;
check if HS_TIMING byte (byte number 185) is 1;
if (HS_TIMING is not 1) report MMC switching to high speed mode failed and return;
change clock divisor value or configure the system clock feeding into uSDHC to generate the
card_clk of around 26MHz or 52MHz according to the CARD_TYPE;
(data transactions like normal peers)
}
disable_mmc_high_speed_mode(void)
{
send CMD6 with argument 0x2B90100;
set BLKCNT field to 1 (block), set BLKSIZE field to 512 (bytes);
send CMD8 to get EXT_CSD value of MMC;
check if HS_TIMING byte (byte number 185) is 0;
if (HS_TIMING is not 0) report the function switch failed and return;
change clock divisor value or configure the system clock feeding into uSDHC to generate the
card_clk of the desired value below 20MHz;
(data transactions like normal peers)
}
```

67.5.4.4 Set MMC Bus Width

```
change_mmc_bus_width(void)
{
send CMD9 to get CSD value of MMC;
check if the value of SPEC_VER field is 4 or above;
if (SPEC_VER value is less than 4) report the MMC does not support multiple bit width and
return;
send CMD6 with argument 0x3B70x00; (8-bit(dual data rate), x=6; 4-bit(dual data rate), x=5;8-
bit, x=2; 4-bit, x=1; 1-bit, x=0)
send CMD13 to wait card ready (busy line released);
(data transactions like normal peers)
}
```

67.5.5 ADMA Operation

67.5.5.1 ADMA1 Operation

```
Set_adma1_descriptor
{
if (to start data transfer) {
// Make sure the address is 4KB align.
Set 'Set' type descriptor;
{
Set Act bits to 01;
Set [31:12] bits data length (byte unit);
}
Set 'Tran' type descriptor;
{
Set Act bits to 10;
Set [31:12] bits address (4KB align);
}
}
else if (to fetch descriptor at non-continuous address) {
Set Act bits to 11;
Set [31:12] bits the next descriptor address (4KB aligned);
}
else { // other types of descriptor
Set Act bits accordingly
}
if (this descriptor is the last one) {
Set End bit to 1;
}
if (to generate interrupt for this descriptor) {
Set Int bit to 1;
}
Set Valid bit to 1;
}
```

67.5.5.2 ADMA2 Operation

```
Set_adma2_descriptor
{
if (to start data transfer) {
// Make sure the address is a 32-bit boundary (lower 2-bit are always '00').
Set higher 32-bit of descriptor for this data transfer initial address;
Set [31:16] bits data length (byte unit);
Set Act bits to '10';
}
else if (to fetch descriptor at non-continuous address) {
Set Act bits to '11';
// Make sure the address is 32-bit boundary (lower 2-bit are always set to '00').
Set higher 32-bit of descriptor for the next descriptor address;
}
else { // other types of descriptor
Set Act bits accordingly
}
if (this descriptor is the last one) {
Set 'End' bit '1';
}
if (to generate interrupt for this descriptor) {
Set 'Int' bit '1';
}
Set the 'Valid' bit to '1';
}
```

67.5.6 Fast Boot Operation

67.5.6.1 Normal fast boot flow

1. Software must configure init_active bit (system control register bit 27) to make sure 74 card clocks are finished.
2. Software must configure the MMC Boot Register (offset 0xc4) bit 6 to 1 (enable boot), and bit 5 to 0 (normal fast boot), and bit 4 to select the ack mode or not. If the data will be sent through DMA mode, the software should configure bit 7 to enable the automatic stop at block gap feature, and configure bit 3-bit 0 to select the ack timeout value according to the SD CLK frequency.
3. Software then needs to configure the Block Attributes Register to set the block size and count. If in DDR fast boot mode, the block size only can be configured to 512 bytes.
4. Software must configure the Protocol control register to set DTW (data transfer width). If in DDR fast boot mode, DTW only can be configured to 4-bit/8-bit dataline mode.
5. Software needs to configure the Command Argument Register to set argument if needed (no need in normal fast boot).
6. Software must configure the Transfer Type Register to start the boot process. In normal boot mode, CMDINX, CMDTYP, RSPTYP, CICEN, CCCEN, AC12EN, BCEN and DMAEN are kept at the default value. DPSEL bit is set to 1, DTDSEL is set to 1, MSBSEL is set to 1.
7. DMAEN should be configured as 0 in polling mode. And if BCEN is configured as 1, it is recommended to configure the number of blocks in the Block Attributes Register to the maximum value. If in DDR fast boot mode, DDR_EN needs to be set to 1.
8. When the step 6 is configured, the boot process will begin. Software needs to poll the data buffer ready status to read the data from the buffer in time. If a boot timeout happens (ack times out or the first data read times out), an interrupt will be triggered, and software must configure MMC Boot Register to bit 6 to 0 to disable boot. This makes CMD high, then after at least 56 clocks, it is ready to begin a normal initialization process.
9. If there is no timeout, software needs to determine when the data read is finished and then configure MMC Boot Register bit 6 to 0 to disable boot. This will make CMD line high and command completed asserted. After at least 56 clocks, it is ready to begin normal initialization process.
10. Reset the host and then can begin the normal process.

67.5.6.2 Alternative fast boot flow

1. Software needs to configure init_active bit (system control register bit 27) to make sure 74 card clocks are finished.
2. Software needs to configure MMC Boot Register (offset 0xc4) bit 6 to 1 (enable boot), and bit 5 to 1 (alternative boot), and bit 4 to select the ack mode or not. If data needs to be sent through DMA mode, then configure bit 7 to enable the automatic stop at block gap feature. Software should also configure bit 3-bit 0 to select the ack timeout value according to the SD clock frequency.
3. Software then needs to configure Block Attributes Register to set the block size and count. If in DDR fast boot mode, the block size only can be configured to 512 bytes.
4. Software needs to configure the Protocol control register to set the DTW (data transfer width). If in ddr fast boot mode, DTW only can be configure to 4-bit/8-bit dataline mode.
5. Software needs to configure Command Argument Register to set argument to 0xFFFFFFFA.
6. Software needs to configure the Transfer Type Register to start the boot process by CMD0 with 0xFFFFFFFA argument . In alternative boot, CMDINX, CMDTYP, RSPTYP, CICEN, CCCEN, AC12EN, BCEN and DMAEN are kept default value. DPSEL bit is set to 1, DTDSEL is set to 1, MSBSEL is set to 1. Note DMAEN should be configured as 0 in polling mode. And if BCEN is configured as 1 in polling mode, it is recommended to configure the block count in the Block Attributes Register to the maximum value. If in DDR fast boot mode, DDR_EN needs to be set to 1.
7. When the step 6 is configured, the boot process will begin. Software needs to poll the data buffer ready status to read the data from the buffer in time. If there is a boot timeout (ack data timeout in 50ms or data timeout in 1s), the host will send out the interrupt and software needs to send CMD0 with reset and then configure the boot enable bit to 0 to stop this process..
8. If there is no time out, software needs to decide when to stop the boot process, and send out the CMD0 with reset and then after the command is completed, configure the MMC Boot Register bit 6 to stop the process. After 8 clocks from the command completion, the slave (card) is ready for the identification step.
9. Reset the host and then begin the normal process.

67.5.6.3 Fast boot application case (in DMA mode)

In the boot application case, because the image destination and the image size are contained in the beginning of the image, it is necessary to switch DMA parameters on the fly during MMC fast boot.

In fast boot, the host can use ADMA2 (Advanced DMA2) with two destinations.

The detail flow is described below:

1. Software needs to configure INIT_ACTIVE bit (system control register bit 27) to make sure 74 card clocks are finished.
2. Software needs to configure the MMC Boot Register (offset 0xc4) bit 6 to 1 (enable boot); and bit 5 to 0 (normal fast boot) or 1 (alternative boot); and bit 4 to select the ack mode or not. In DMA mode, configure bit 7 to 1 to enable the automatic stop at block gap feature. Also configure bits[31-16] to set the (BLK_CNT - VALUE1).Here VALUE1 is the value of the block count that needs to transfer the first time, so that that the hostwill stop at the block gap when the uSDHC controller gets VAULE1 blocks from the device. Also configure bits[3-0] to select the ack timeout value according to the SD clock frequency.
3. Software then needs to configure the Block Attributes Register to set block size and count. If in DDR fast boot mode, the block size only can be configured to 512 bytes. In DMA mode, it is recommended to set the block count (BLK_CNT) to the max value (16'hffff).
4. Software needs to configure Protocol Control Register to set DTW (data transfer width). If in DDR fast boot mode, the DTW only can be configured to 4-bit/8-bit dataline mode.
5. Software enbable ADMA2 by configuring Protocol Control Register bits [9-8].
6. Software need to set at least three pairs ADMA2 descriptor in boot memory (ie, in IRAM, at least 6 word). The first pair descriptor define the start address (ie, IRAM)and data length(ie,512byte*VALUE1) of fisrt part boot code. Software also need to set the second pair descriptor, the second start address (any value that is writeable), data length is suggest to set 1~2word(record as VALUE2). Note: the second couple desc also transfer useful data even at lease 1 word. Because our ADMA2 can't support 0 data_length data transfer descriptor.
7. Software needs to configure Command Argument Register to set argument to 0xFFFFFFFA in alternative fast boot, and don't need set in normal fast boot.
8. Software needs to configure Transfer Type Register to start the boot process . CMDINX, CMDTYP, RSPTYP, CICEN, CCCEN, AC12EN, BCEN and DMAEN are kept default value. DPSEL bit is set to 1, DTDSEL is set to 1, MSBSEL is set to 1. DMAEN is configured as 1 in DMA mode. And if BCEN is configured as 1, better to configure blk no in Bock Attributes Register to the max value. And if in ddr fast boot mode, DDR_EN need to be set to 1.
9. When the step 8 is configured, boot process will begin, the first VALUE1 block number data has transfer. Software need to polling TC bit (bit1 in Interrupt Status Register) to determine first transfer is end. Also software need to polling BGE bit (bit2 in Interrupt Status Register) to determine if first transfer stop at block gap.
10. When TC, BGE bit is 1, . SW can analyzes the first code of VALUE1 block, initializes the new memory device, if required, and sets the third pair of descriptors to

define the start address and length of the remaining part of boot code(VALUE3 the remain boot code block). Remeber set the last descriptor with END.

11. Software needs to configure MMC Boot Register (offset 0xc4) again. Set bit 6 to 1(enable boot); and bit 5 to 0(normal fast boot), to 1(alternative boot); and bit 4 to select the ack mode or not. In DMA mode, configure bit 7 to 1 for enable automatically stop at block gap feature. Also configure bit31-bit16 to set the *(BLK_CNT - (VALUE1+1+VALUE3))*, that host will stop at block gap when *the uSDHC controller gets (VALUE1+1+VALUE3)) blocks from device totally include the blocks received in step 9*. And need to configure bit 3-bit0 to select the ack timeout value according to the sd clk frequence. Please note, Software doesn't need to configure the *BLK_CNT* again, because it's counted down automatically by the uSDHC controller.
12. Software needs to clear TC and BGE bit. And software needs to clear SABGREQ(bit 16 in Protocol control register), and set CREQ(bit17 Protocol control register) to 1 to resume the data transfer. Host will transfer the VALUE2 and VALUE3 data to the destination that is set by descriptor.
13. Software need to polling BGE bit to determine if the fast boot is over.

Note:

1. When ADMA boot flow is started, for uSDHC, it is like a normal ADMA read operation. So setting ADMA2 descriptor as the normal ADMA2 transfer.
2. Need a few words length memory to keep descriptor.
3. For the 1~2 word data in second descriptor setting, it is the useful data, so software need to deal the data due to the application case.

67.6 Commands for MMC/SD/SDIO

A table containing the list of commands for the MMC/SD/SDIO cards can be found here.

Refer to the corresponding specifications for more details about the command information.

There are four kinds of commands defined to control the MultiMediaCard:

1. broadcast commands (bc), no response.
2. broadcast commands with response (bcr), response from all cards simultaneously.
3. addressed (point-to-point) commands (ac), no data transfer on the DATA.
4. addressed (point-to-point) data transfer commands (adtc).

Response: a response is a token which is sent from the card to the host as an answner to a previously received command. A response is transferred serially on the CMD line.

Table 67-6. Commands for MMC/SD/SDIO Cards

CMD INDEX	Type	Argument	Response type	Abbreviation	Description
CMD0	bc	[31:0] stuff bits	-	GO_IDLE_STATE	Resets all MMC and SD memory cards to idle state.
CMD1	bcr	[31:0] OCR without busy	R3	SEND_OP_COND	Asks all MMC and SD Memory cards in idle state to send their operation conditions register contents in the response on the CMD line.
CMD2	bcr	[31:0] stuff bits	R2	ALL_SEND_CID	Asks all cards to send their CID numbers on the CMD line.
CMD3[1]	ac	[31:6] RCA [15:0] stuff bits	R1 R6 (SDIO)	SET/ SEND_RELATIVE_ADDR	Assigns relative address to the card.
CMD4	bc	[31:0] DSR [15:0] stuff bits	-	SET_DSR	Programs the DSR of all cards.
CMD5	bc	[31:0] OCR without busy	R4	IO_SEND_OP_COND	Asks all SDIO cards in idle state to send their operation conditions register contents in the response on the CMD line.
CMD6[2]	adtc	[31] Mode 0: Check function 1: Switch function [30:8] Reserved for function groups 6 ~ 3 (All 0 or 0xFFFF) [7:4] Function group1 for command system [3:0] Function group2 for access mode	R1	SWITCH_FUNC	Checks switch ability (mode 0) and switch card function (mode 1). Refer to "SD Physical Specification V1.1" for more details.
CMD6[3]	ac	[31:26] Set to 0 [25:24] Access [23:16] Index [15:8] Value [7:3] Set to 0 [2:0] Cmd Set	R1b	SWITCH	Switches the mode of operation of the selected card or modifies the EXT_CSD registers. Refer to "The MultiMediaCard System Specification Version 4.0 Final draft 2" for more details.
CMD7	ac	[31:6] RCA [15:0] stuff bits	R1b	SELECT/ DESELECT_CARD	Toggles a card between the stand-by and transfer states or between the programming and disconnect states. In both cases, the card is selected by its own relative address and gets deselected by any other address. Address 0 deselects all.
CMD8	adtc	[31:0] stuff bits	R1	SEND_EXT_CSD	The card sends its EXT_CSD register as a block of data, with a block size of 512 bytes.

Table continues on the next page...

Table 67-6. Commands for MMC/SD/SDIO Cards (continued)

CMD INDEX	Type	Argument	Response type	Abbreviation	Description
CMD9	ac	[31:6] RCA [15:0] stuff bits	R2	SEND_CSD	Addressed card sends its card-specific data (CSD) on the CMD line.
CMD10	ac	[31:6] RCA [15:0] stuff bits	R2	SEND_CID	Addressed card sends its card-identification (CID) on the CMD line.
CMD11	adtc	[31:0] data address	R1	READ_DAT_UNTIL_STOP	Reads data stream from the card, starting at the given address, until a STOP_TRANSMISSION follows.
CMD12	ac	[31:0] stuff bits	R1b	STOP_TRANSMISSION	Forces the card to stop transmission.
CMD13	ac	[31:6] RCA [15:0] stuff bits	R1	SEND_STATUS	Addressed card sends its status register.
CMD14	Reserved				
CMD15	ac	[31:6] RCA [15:0] stuff bits	-	GO_INACTIVE_STATE	Sets the card to inactive state in order to protect the card stack against communication breakdowns.
CMD16	ac	[31:0] block length	R1	SET_BLOCKLEN	Sets the block length (in bytes) for all following block commands (read and write). Default block length is specified in the CSD.
CMD17	adtc	[31:0] data address	R1	READ_SINGLE_BLOCK	Reads a block of the size selected by the SET_BLOCKLEN command.
CMD18	adtc	[31:0] data address	R1	READ_MULTIPLE_BLOCK	Continuously transfers data blocks from card to host until interrupted by a stop command.
CMD19	Reserved				
CMD20	adtc	[31:0] data address	R1	WRITE_DAT_UNTIL_STOP	Writes data stream from the host, starting at the given address, until a STOP_TRANSMISION follows.
CMD21-23	Reserved				
CMD24	adtc	[31:0] data address	R1	WRITE_BLOCK	Writes a block of the size selected by the SET_BLOCKLEN command.
CMD25	adtc	[31:0] data address	R1	WRITE_MULTIPLE_BLOCK	Continuously writes blocks of data until a STOP_TRANSMISSION follows.
CMD26	adtc	[31:0] stuff bits	R1	PROGRAM_CID	Programming of the card identification register. This command shall be issued only once per card. The card contains hardware to prevent this operation after the first programming. Normally this command is reserved for the manufacturer.
CMD27	adtc	[31:0] stuff bits	R1	PROGRAM_CSD	Programming of the programmable bits of the CSD.

Table continues on the next page...

Table 67-6. Commands for MMC/SD/SDIO Cards (continued)

CMD INDEX	Type	Argument	Response type	Abbreviation	Description
CMD28	ac	[31:0] data address	R1b	SET_WRITE_PROT	If the card has write protection features, this command sets the write protection bit of the addressed group. The properties of write protection are coded in the card specific data (WP_GRP_SIZE).
CMD29	ac	[31:0] data address	R1b	CLR_WRITE_PROT	If the card provides write protection features, this command clears the write protection bit of the addressed group.
CMD30	adtc	[31:0] write protect data address	R1	SEND_WRITE_PROT	If the card provides write protection features, this command asks the card to send the status of the write protection bits.
CMD31	Reserved				
CMD32	ac	[31:0] data address	R1	TAG_SECTOR_START	Sets the address of the first sector of the erase group.
CMD33	ac	[31:0] data address	R1	TAG_SECTOR_END	Sets the address of the last sector in a continuous range within the selection of a single sector to be selected for erase.
CMD34	ac	[31:0] data address	R1	UNTAG_SECTOR	Removes one previously selected sector from the erase selection.
CMD35	ac	[31:0] data address	R1	TAG_ERASE_GROUP_START	Sets the address of the first erase group within a range to be selected for erase.
CMD36	ac	[31:0] data address	R1	TAG_ERASE_GROUP_END	Sets the address of the last erase group within a continuous range to be selected for erase.
CMD37	ac	[31:0] data address	R1	UNTAG_ERASE_GROUP	Removes one previously selected erase group from the erase selection.
CMD38	ac	[31:0] stuff bits	R1b	ERASE	Erase all previously selected sectors.
CMD39	ac	[31:0] RCA [15] register write flag [14:8] register address [7:0] register data	R4	FAST_IO	Used to write and read 8-bit (register) data fields. The command addresses a card, and a register, and provides the data for writing if the write flag is set. The R4 response contains data read from the address register. This command accesses application dependent registers which are not defined in the MMC standard.
CMD40	bcr	[31:0] stuff bits	R5	GO_IRQ_STATE	Sets the system into interrupt mode.
CMD41	Reserved				

Table continues on the next page...

Table 67-6. Commands for MMC/SD/SDIO Cards (continued)

CMD INDEX	Type	Argument	Response type	Abbreviation	Description
CDM42	adtc	[31:0] stuff bits	R1b	LOCK_UNLOCK	Used to set/reset the password or lock/unlock the card. The size of the data block is set by the SET_BLOCK_LEN command.
CMD43~51	Reserved				
CMD52	ac	[31:0] stuff bits	R5	IO_RW_DIRECT	Access a single register within the total 128k of register space in any I/O function.
CMD53	ac	[31:0] stuff bits	R5	IO_RW_EXTENDED	Accesses a multiple I/O register with a single command. Allows the reading or writing of a large number of I/O registers.
CMD54	Reserved				
CMD55	ac	[31:16] RCA [15:0] stuff bits	R1	APP_CMD	Indicates to the card that the next command is an application specific command rather that a standard command.
CMD56	adtc	[31:1] stuff bits [0]: RD/WR	R1b	GEN_CMD	Used either to transfer a data block to the card or to get a data block from the card for general purpose / application specific commands. The size of the data block is set by the SET_BLOCK_LEN command.
CMD57-59	Reserved				
CMD60	adtc	[31] WR [30:24] stuff bits [23:16] address [15:8] stuff bits [7:0] byte count	R1b	RW_MULTIPLE_REGISTER	These registers are used to control the behavior of the device and to retrieve status information regarding the operation of the device. All Status and Control registers are WORD (32-bit) in size and are WORD aligned. CMD60 shall be used to read and write these registers.
CMD61	adtc	[31] WR [30:16] stuff bits [15:0] data unit count	R1b	RW_MULTIPLE_BLOCK	The host issues a RW_MULTIPLE_BLOCK (CMD61) to begin the data transfer.
CMD62-63	Reserved				
ACMD6[4]	ac	[31:2] stuff bits [1:0] bus width	R1	SET_BUS_WIDTH	Defines the data bus width ('00'=1bit or '10'=4bit bus) to be used for data transfer. The allowed data bus widths are given in SCR register.
ACMD13[4]	adtc	[31:0] stuff bits	R1	SD_STATUS	Send the SD Memory Card status.
ACMD22[4]	adtc	[31:0] stuff bits	R1	SEND_NUM_WR_SECTORS	Send the number of the written sectors (without errors). Responds with 32-bit plus the CRC data block.

Table continues on the next page...

Table 67-6. Commands for MMC/SD/SDIO Cards (continued)

CMD INDEX	Type	Argument	Response type	Abbreviation	Description
ACMD23[4]	ac	[31:23] stuff bits [22:0] Number of blocks	R1	SET_WR_BLK_ERASE_COUNT	Set the number of write blocks to be pre-erased before writing (to be used for fast Multiple Block WR command). "1"=default(one write block).
ACMD41[4]	bcr	[31:0] OCR	R3	SD_APP_OP_COND	Asks the accessed card to send its operating condition register (OCR) contents in the response on the CMD line.
ACMD42[4]	ac	[31:1] stuff bits [0] set_cd	R1	SET_CLR_CARD_DETECT	Connect(1)/Disconnect(0) the 50KOhm pull-up resistor on CD_B/DATA3 of the card.
ACMD51[4]	adtc	[31:0] stuff bits	R1	SEND_SCR	Reads the SD Configuration Register (SCR).

1. CMD3 differs for MMC and SD cards. For MMC cards, it is referred to as SET_RELATIVE_ADDR, with a response type of R1. For SD cards, it is referred to as SEND_RELATIVE_ADDR, with a response type of R6 (with RCA inside).
2. CMD6 differs completely between high speed MMC cards and high speed SD cards. Command SWITCH_FUNC is for high speed SD cards.
3. Command SWITCH is for high speed MMC cards . The Index field can contain any value from 0-255, but only values 0-191 are valid. If the Index value is in the 192-255 range the card does not perform any modification and the SWITCH_ERROR status bit in the EXT_CSD register is set. The Access Bits are shown in Table 2.
4. ACMDs shall be preceded with the APP_CMD command. (Commands listed are used for SD only, other SD commands not listed are not supported on this module).

The Access Bits for the EXT_CSD Access Modes are shown below.

Table 67-7. EXT_CSD Access Modes

Bits	Access Name	Operation
00	Command Set	The command set is changed according to the Cmd Set field of the argument
01	Set Bits	The bits in the pointed byte are set, according to the 1 bits in the Value field.
10	Clear Bits	The bits in the pointed byte are cleared, according to the 1 bits in the Value field.
11	Write Byte	The Value field is written into the pointed byte.

67.7 Software Restrictions

67.7.1 Initialization Active

The driver cannot set INITA bit in System Control register when any of the command line or data lines is active, so the driver must ensure both CDIHB and CIHB bits are cleared.

67.7.2 Software Polling Procedure

For polling read or write, once the software begins a buffer read or write, it must access exactly the number of times as the values set in the Watermark Level Register; moreover, if the block size is not a multiple of the value in Watermark Level Register (read and write respectively), the software must access exactly the remaining number of words at the end of each block.

For example, for a read operation, if the RD_WML is 4, indicating the watermark level is 16 bytes, block size is 40 bytes, and the block number is 2, then the access times for the burst sequence in the whole transfer process must be 4, 4, 2, 4, 4, 2.

67.7.3 Suspend Operation

In order to suspend the data transfer, the software must inform uSDHC that the suspend command is successfully accepted. To achieve this, after the Suspend command is accepted by the SDIO card, software must send another normal command marked as suspend command (CMDTYP bits set as '01') to inform uSDHC that the transfer is suspended.

If software needs to resume the suspended trasnfer, it should read the value in BLKCNT register to save the remaining number of blocks before sending the normal command marked as suspend, otherwise on sending such 'suspend' command, uSDHC will regard the current transfer is aborted and change BLKCNT register to its original value, instead of keeping the remained number of blocks.

67.7.4 Data Length Setting

For either ADMA (ADMA1 or ADMA2) transfer, the data in the data buffer must be word aligned, so the data length set in the descriptor must be a multiple of 4.

67.7.5 (A)DMA Address Setting

To configure ADMA1/ADMA2/DMA address register, when TC bit is set, the register will always update itself with the internal address value to support dynamic address synchronization, so the software must ensure that the TC bit is cleared prior to configuring ADMA1/ADMA2/DMA address register.

67.7.6 Data Port Access

Data Port does not support parallel access. For example, during an external DMA access, it is not allowed to write any data to the Data Port by CPU; or during a CPU read operation, it is also prohibited to write any data to the Data Port, by either CPU or external DMA. Otherwise the data would be corrupted inside the uSDHC buffer.

67.7.7 Change Clock Frequency

uSDHC does not automatically gate off the card clock when the Host Driver changes the clock frequency. To prevent possible glitch on the card clock, clear the FRC_SDCLK_ON bit when changing clock divisor value(SDCLKFS or DVS in System Control Register) or setting RSTA bit.

Also before changing the clock divisor value, Host Driver should make sure the SDSTB bit is high.

67.7.8 Multi-block Read

For pre-defined multi-block read operation, i.e., the number of blocks to read has been defined by previous CMD23 for MMC, or pre-defined number of blocks in CMD53 for SDIO/SDCombo, or whatever multi-block read without abort command at card side, an abort command, either automatic or manual CMD12/CMD52, is still required by uSDHC after the pre-defined number of blocks are done, to drive the internal state machine to idle mode.

In this case, the card may not respond to this extra abort command and uSDHC will get Response Timeout. It is recommended to manually send an abort command with RSPTYP[1:0] both bits cleared.

67.8 uSDHC Memory Map/Register Definition

This section includes the module memory map and detailed descriptions of all registers.

See the table below for the register memory map for the uSDHC. All these registers only support 32-bit accesses.

NOTE

The uSDHC registers are 32-bit wide and only support 32-bit access.

uSDHC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
219_0000	DMA System Address (uSDHC1_DS_ADDR)	32	R/W	0000_0000h	67.8.1/5629
219_0004	Block Attributes (uSDHC1_BLK_ATT)	32	R/W	0000_0000h	67.8.2/5629
219_0008	Command Argument (uSDHC1_CMD_ARG)	32	R/W	0000_0000h	67.8.3/5630
219_000C	Command Transfer Type (uSDHC1_CMD_XFR_TYP)	32	R/W	0000_0000h	67.8.4/5631
219_0010	Command Response0 (uSDHC1_CMD_RSP0)	32	R	0000_0000h	67.8.5/5634
219_0014	Command Response1 (uSDHC1_CMD_RSP1)	32	R	0000_0000h	67.8.6/5635
219_0018	Command Response2 (uSDHC1_CMD_RSP2)	32	R	0000_0000h	67.8.7/5635
219_001C	Command Response3 (uSDHC1_CMD_RSP3)	32	R	0000_0000h	67.8.8/5636
219_0020	Data Buffer Access Port (uSDHC1_DATA_BUFF_ACC_PORT)	32	R/W	0000_0000h	67.8.9/5637
219_0024	Present State (uSDHC1_PRES_STATE)	32	R	0000_0000h	67.8.10/ 5637
219_0028	Protocol Control (uSDHC1_PROT_CTRL)	32	R/W	0880_0020h	67.8.11/ 5643
219_002C	System Control (uSDHC1_SYS_CTRL)	32	R/W	0080_800Fh	67.8.12/ 5648
219_0030	Interrupt Status (uSDHC1_INT_STATUS)	32	w1c	0000_0000h	67.8.13/ 5651
219_0034	Interrupt Status Enable (uSDHC1_INT_STATUS_EN)	32	R/W	0000_0000h	67.8.14/ 5657
219_0038	Interrupt Signal Enable (uSDHC1_INT_SIGNAL_EN)	32	R/W	0000_0000h	67.8.15/ 5660
219_003C	Auto CMD12 Error Status (uSDHC1_AUTOCMD12_ERR_STATUS)	32	R	0000_0000h	67.8.16/ 5662
219_0040	Host Controller Capabilities (uSDHC1_HOST_CTRL_CAP)	32	R	07F3_0000h	67.8.17/ 5666
219_0044	Watermark Level (uSDHC1_WTMK_LVL)	32	R/W	0810_0810h	67.8.18/ 5668
219_0048	Mixer Control (uSDHC1_MIX_CTRL)	32	R/W	8000_0000h	67.8.19/ 5669

Table continues on the next page...

uSDHC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
219_0050	Force Event (uSDHC1_FORCE_EVENT)	32	W (always reads 0)	0000_0000h	67.8.20/ 5671
219_0054	ADMA Error Status Register (uSDHC1_ADMA_ERR_STATUS)	32	R	0000_0000h	67.8.21/ 5673
219_0058	ADMA System Address (uSDHC1_ADMA_SYS_ADDR)	32	R/W	0000_0000h	67.8.22/ 5676
219_0060	DLL (Delay Line) Control (uSDHC1_DLL_CTRL)	32	R/W	0000_0200h	67.8.23/ 5677
219_0064	DLL Status (uSDHC1_DLL_STATUS)	32	R	0000_0000h	67.8.24/ 5679
219_0068	CLK Tuning Control and Status (uSDHC1_CLK_TUNE_CTRL_STATUS)	32	R/W	0000_0000h	67.8.25/ 5680
219_00C0	Vendor Specific Register (uSDHC1_VEND_SPEC)	32	R/W	2000_7809h	67.8.26/ 5682
219_00C4	MMC Boot Register (uSDHC1_MMC_BOOT)	32	R/W	0000_0000h	67.8.27/ 5685
219_00C8	Vendor Specific 2 Register (uSDHC1_VEND_SPEC2)	32	R/W	0000_0006h	67.8.28/ 5686
219_4000	DMA System Address (uSDHC2_DS_ADDR)	32	R/W	0000_0000h	67.8.1/5629
219_4004	Block Attributes (uSDHC2_BLK_ATT)	32	R/W	0000_0000h	67.8.2/5629
219_4008	Command Argument (uSDHC2_CMD_ARG)	32	R/W	0000_0000h	67.8.3/5630
219_400C	Command Transfer Type (uSDHC2_CMD_XFR_TYP)	32	R/W	0000_0000h	67.8.4/5631
219_4010	Command Response0 (uSDHC2_CMD_RSP0)	32	R	0000_0000h	67.8.5/5634
219_4014	Command Response1 (uSDHC2_CMD_RSP1)	32	R	0000_0000h	67.8.6/5635
219_4018	Command Response2 (uSDHC2_CMD_RSP2)	32	R	0000_0000h	67.8.7/5635
219_401C	Command Response3 (uSDHC2_CMD_RSP3)	32	R	0000_0000h	67.8.8/5636
219_4020	Data Buffer Access Port (uSDHC2_DATA_BUFF_ACC_PORT)	32	R/W	0000_0000h	67.8.9/5637
219_4024	Present State (uSDHC2_PRES_STATE)	32	R	0000_0000h	67.8.10/ 5637
219_4028	Protocol Control (uSDHC2_PROT_CTRL)	32	R/W	0880_0020h	67.8.11/ 5643
219_402C	System Control (uSDHC2_SYS_CTRL)	32	R/W	0080_800Fh	67.8.12/ 5648
219_4030	Interrupt Status (uSDHC2_INT_STATUS)	32	w1c	0000_0000h	67.8.13/ 5651
219_4034	Interrupt Status Enable (uSDHC2_INT_STATUS_EN)	32	R/W	0000_0000h	67.8.14/ 5657
219_4038	Interrupt Signal Enable (uSDHC2_INT_SIGNAL_EN)	32	R/W	0000_0000h	67.8.15/ 5660
219_403C	Auto CMD12 Error Status (uSDHC2_AUTOCMD12_ERR_STATUS)	32	R	0000_0000h	67.8.16/ 5662

Table continues on the next page...

uSDHC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
219_4040	Host Controller Capabilities (uSDHC2_HOST_CTRL_CAP)	32	R	07F3_0000h	67.8.17/ 5666
219_4044	Watermark Level (uSDHC2_WTMK_LVL)	32	R/W	0810_0810h	67.8.18/ 5668
219_4048	Mixer Control (uSDHC2_MIX_CTRL)	32	R/W	8000_0000h	67.8.19/ 5669
219_4050	Force Event (uSDHC2_FORCE_EVENT)	32	W (always reads 0)	0000_0000h	67.8.20/ 5671
219_4054	ADMA Error Status Register (uSDHC2_ADMA_ERR_STATUS)	32	R	0000_0000h	67.8.21/ 5673
219_4058	ADMA System Address (uSDHC2_ADMA_SYS_ADDR)	32	R/W	0000_0000h	67.8.22/ 5676
219_4060	DLL (Delay Line) Control (uSDHC2_DLL_CTRL)	32	R/W	0000_0200h	67.8.23/ 5677
219_4064	DLL Status (uSDHC2_DLL_STATUS)	32	R	0000_0000h	67.8.24/ 5679
219_4068	CLK Tuning Control and Status (uSDHC2_CLK_TUNE_CTRL_STATUS)	32	R/W	0000_0000h	67.8.25/ 5680
219_40C0	Vendor Specific Register (uSDHC2_VEND_SPEC)	32	R/W	2000_7809h	67.8.26/ 5682
219_40C4	MMC Boot Register (uSDHC2_MMC_BOOT)	32	R/W	0000_0000h	67.8.27/ 5685
219_40C8	Vendor Specific 2 Register (uSDHC2_VEND_SPEC2)	32	R/W	0000_0006h	67.8.28/ 5686
219_8000	DMA System Address (uSDHC3_DS_ADDR)	32	R/W	0000_0000h	67.8.1/5629
219_8004	Block Attributes (uSDHC3_BLK_ATT)	32	R/W	0000_0000h	67.8.2/5629
219_8008	Command Argument (uSDHC3_CMD_ARG)	32	R/W	0000_0000h	67.8.3/5630
219_800C	Command Transfer Type (uSDHC3_CMD_XFR_TYP)	32	R/W	0000_0000h	67.8.4/5631
219_8010	Command Response0 (uSDHC3_CMD_RSP0)	32	R	0000_0000h	67.8.5/5634
219_8014	Command Response1 (uSDHC3_CMD_RSP1)	32	R	0000_0000h	67.8.6/5635
219_8018	Command Response2 (uSDHC3_CMD_RSP2)	32	R	0000_0000h	67.8.7/5635
219_801C	Command Response3 (uSDHC3_CMD_RSP3)	32	R	0000_0000h	67.8.8/5636
219_8020	Data Buffer Access Port (uSDHC3_DATA_BUFF_ACC_PORT)	32	R/W	0000_0000h	67.8.9/5637
219_8024	Present State (uSDHC3_PRES_STATE)	32	R	0000_0000h	67.8.10/ 5637
219_8028	Protocol Control (uSDHC3_PROT_CTRL)	32	R/W	0880_0020h	67.8.11/ 5643
219_802C	System Control (uSDHC3_SYS_CTRL)	32	R/W	0080_800Fh	67.8.12/ 5648
219_8030	Interrupt Status (uSDHC3_INT_STATUS)	32	w1c	0000_0000h	67.8.13/ 5651

Table continues on the next page...

uSDHC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
219_8034	Interrupt Status Enable (uSDHC3_INT_STATUS_EN)	32	R/W	0000_0000h	67.8.14/ 5657
219_8038	Interrupt Signal Enable (uSDHC3_INT_SIGNAL_EN)	32	R/W	0000_0000h	67.8.15/ 5660
219_803C	Auto CMD12 Error Status (uSDHC3_AUTOCMD12_ERR_STATUS)	32	R	0000_0000h	67.8.16/ 5662
219_8040	Host Controller Capabilities (uSDHC3_HOST_CTRL_CAP)	32	R	07F3_0000h	67.8.17/ 5666
219_8044	Watermark Level (uSDHC3_WTMK_LVL)	32	R/W	0810_0810h	67.8.18/ 5668
219_8048	Mixer Control (uSDHC3_MIX_CTRL)	32	R/W	8000_0000h	67.8.19/ 5669
219_8050	Force Event (uSDHC3_FORCE_EVENT)	32	W (always reads 0)	0000_0000h	67.8.20/ 5671
219_8054	ADMA Error Status Register (uSDHC3_ADMA_ERR_STATUS)	32	R	0000_0000h	67.8.21/ 5673
219_8058	ADMA System Address (uSDHC3_ADMA_SYS_ADDR)	32	R/W	0000_0000h	67.8.22/ 5676
219_8060	DLL (Delay Line) Control (uSDHC3_DLL_CTRL)	32	R/W	0000_0200h	67.8.23/ 5677
219_8064	DLL Status (uSDHC3_DLL_STATUS)	32	R	0000_0000h	67.8.24/ 5679
219_8068	CLK Tuning Control and Status (uSDHC3_CLK_TUNE_CTRL_STATUS)	32	R/W	0000_0000h	67.8.25/ 5680
219_80C0	Vendor Specific Register (uSDHC3_VEND_SPEC)	32	R/W	2000_7809h	67.8.26/ 5682
219_80C4	MMC Boot Register (uSDHC3_MMC_BOOT)	32	R/W	0000_0000h	67.8.27/ 5685
219_80C8	Vendor Specific 2 Register (uSDHC3_VEND_SPEC2)	32	R/W	0000_0006h	67.8.28/ 5686
219_C000	DMA System Address (uSDHC4_DS_ADDR)	32	R/W	0000_0000h	67.8.1/5629
219_C004	Block Attributes (uSDHC4_BLK_ATT)	32	R/W	0000_0000h	67.8.2/5629
219_C008	Command Argument (uSDHC4_CMD_ARG)	32	R/W	0000_0000h	67.8.3/5630
219_C00C	Command Transfer Type (uSDHC4_CMD_XFR_TYP)	32	R/W	0000_0000h	67.8.4/5631
219_C010	Command Response0 (uSDHC4_CMD_RSP0)	32	R	0000_0000h	67.8.5/5634
219_C014	Command Response1 (uSDHC4_CMD_RSP1)	32	R	0000_0000h	67.8.6/5635
219_C018	Command Response2 (uSDHC4_CMD_RSP2)	32	R	0000_0000h	67.8.7/5635
219_C01C	Command Response3 (uSDHC4_CMD_RSP3)	32	R	0000_0000h	67.8.8/5636
219_C020	Data Buffer Access Port (uSDHC4_DATA_BUFF_ACC_PORT)	32	R/W	0000_0000h	67.8.9/5637
219_C024	Present State (uSDHC4_PRES_STATE)	32	R	0000_0000h	67.8.10/ 5637

Table continues on the next page...

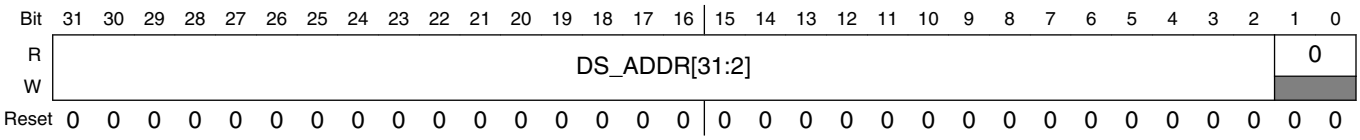
uSDHC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
219_C028	Protocol Control (uSDHC4_PROT_CTRL)	32	R/W	0880_0020h	67.8.11/ 5643
219_C02C	System Control (uSDHC4_SYS_CTRL)	32	R/W	0080_800Fh	67.8.12/ 5648
219_C030	Interrupt Status (uSDHC4_INT_STATUS)	32	w1c	0000_0000h	67.8.13/ 5651
219_C034	Interrupt Status Enable (uSDHC4_INT_STATUS_EN)	32	R/W	0000_0000h	67.8.14/ 5657
219_C038	Interrupt Signal Enable (uSDHC4_INT_SIGNAL_EN)	32	R/W	0000_0000h	67.8.15/ 5660
219_C03C	Auto CMD12 Error Status (uSDHC4_AUTOCMD12_ERR_STATUS)	32	R	0000_0000h	67.8.16/ 5662
219_C040	Host Controller Capabilities (uSDHC4_HOST_CTRL_CAP)	32	R	07F3_0000h	67.8.17/ 5666
219_C044	Watermark Level (uSDHC4_WTMK_LVL)	32	R/W	0810_0810h	67.8.18/ 5668
219_C048	Mixer Control (uSDHC4_MIX_CTRL)	32	R/W	8000_0000h	67.8.19/ 5669
219_C050	Force Event (uSDHC4_FORCE_EVENT)	32	W (always reads 0)	0000_0000h	67.8.20/ 5671
219_C054	ADMA Error Status Register (uSDHC4_ADMA_ERR_STATUS)	32	R	0000_0000h	67.8.21/ 5673
219_C058	ADMA System Address (uSDHC4_ADMA_SYS_ADDR)	32	R/W	0000_0000h	67.8.22/ 5676
219_C060	DLL (Delay Line) Control (uSDHC4_DLL_CTRL)	32	R/W	0000_0200h	67.8.23/ 5677
219_C064	DLL Status (uSDHC4_DLL_STATUS)	32	R	0000_0000h	67.8.24/ 5679
219_C068	CLK Tuning Control and Status (uSDHC4_CLK_TUNE_CTRL_STATUS)	32	R/W	0000_0000h	67.8.25/ 5680
219_C0C0	Vendor Specific Register (uSDHC4_VEND_SPEC)	32	R/W	2000_7809h	67.8.26/ 5682
219_C0C4	MMC Boot Register (uSDHC4_MMC_BOOT)	32	R/W	0000_0000h	67.8.27/ 5685
219_C0C8	Vendor Specific 2 Register (uSDHC4_VEND_SPEC2)	32	R/W	0000_0006h	67.8.28/ 5686

67.8.1 DMA System Address (uSDHCx_DS_ADDR)

This register contains the physical system memory address used for DMA transfers.

Address: Base address + 0h offset



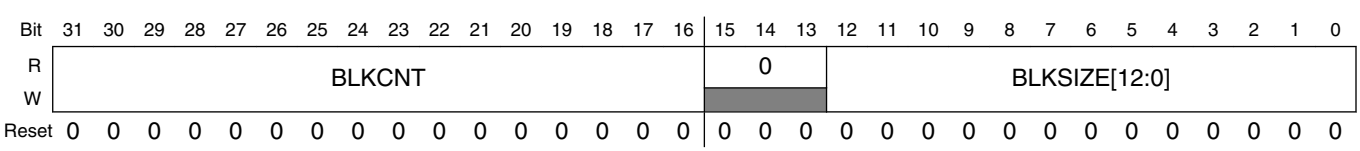
uSDHCx_DS_ADDR field descriptions

Field	Description
31–2 DS_ADDR[31:2]	DMA System Address: This register contains the 32-bit system memory address for a DMA transfer. Since the address must be word (4 bytes) aligned, the least 2 bits are reserved, always 0. When the uSDHC stops a DMA transfer, this register points to the system address of the next contiguous data position. It can be accessed only when no transaction is executing (i.e. after a transaction has stopped). Read operation during transfers may return an invalid value. The Host Driver shall initialize this register before starting a DMA transaction. After DMA has stopped, the system address of the next contiguous data position can be read from this register. This register is protected during a data transfer. When data lines are active, write to this register is ignored. The Host driver shall wait, until the DLA bit in the Present State register is cleared, before writing to this register. The uSDHC internal DMA does not support a virtual memory system. It only supports continuous physical memory access. And due to AHB burst limitations, if the burst must cross the 1 KB boundary, uSDHC will automatically change SEQ burst type to NSEQ. Since this register supports dynamic address reflecting, when TC bit is set, it automatically alters the value of internal address counter, so SW cannot change this register when TC bit is set. Such restriction is also listed in Software Restrictions .
Reserved	This read-only field is reserved and always has the value 0.

67.8.2 Block Attributes (uSDHCx_BLK_ATT)

This register is used to configure the number of data blocks and the number of bytes in each block.

Address: Base address + 4h offset



uSDHCx_BLK_ATT field descriptions

Field	Description
31–16 BLKCNT	Blocks Count For Current Transfer: This register is enabled when the Block Count Enable bit in the Transfer Mode register is set to 1 and is valid only for multiple block transfers. For single block transfer, this register will always read as 1. The Host Driver shall set this register to a value between 1 and the maximum block count. The uSDHC 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 being 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 saving transfer content as a result of a Suspend command, the number of blocks yet to be transferred can be determined by reading this register. The reading of this register should be applied after transfer is paused by stop at block gap operation and before sending the command marked as suspend. This is because when Suspend command is sent out, uSDHC will regard the current transfer is aborted and change BLKCNT register back to its original value instead of keeping the dynamical indicator of remained block count. When restoring transfer content prior to issuing a Resume command, the Host Driver shall restore the previously saved block count. **NOTE:** Although the BLKCNT field is 0 after reset, the read of reset value is 0x1. This is because when MSBSEL bit is indicating a single block transfer, the read value of BLKCNT is always 1. FFFF 65535 blocks 0002 2 blocks 0001 1 block 0000 Stop Count
15–13 Reserved	This read-only field is reserved and always has the value 0.
BLKSIZE[12:0]	Transfer Block Size: This register specifies the block size for block data transfers. Values ranging from 1 byte up to the maximum buffer size can be set. It can be accessed only when no transaction is executing (i.e. after a transaction has stopped). Read operations during transfers may return an invalid value, and write operations will be ignored. 1000 4096 Bytes 800 2048 Bytes 200 512 Bytes 1FF 511 Bytes 004 4 Bytes 003 3 Bytes 002 2 Bytes 001 1 Byte 000 No data transfer

67.8.3 Command Argument (uSDHCx_CMD_ARG)

This register contains the SD/MMC Command Argument.

.

Address: Base address + 8h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMDARG[31:0]																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

uSDHCx_CMD_ARG field descriptions

Field	Description
CMDARG[31:0]	Command Argument: The SD/MMC Command Argument is specified as bits 39-8 of the Command Format in the SD or MMC Specification.This register is write protected when the Command Inhibit (CMD) bit in the Present State register is set.

67.8.4 Command Transfer Type (uSDHCx_CMD_XFR_TYP)

This register is used to control the operation of data transfers. The Host Driver shall set this register before issuing a command followed by a data transfer, or before issuing a Resume command. To prevent data loss, the uSDHC prevents writing to the bits, that are involved in the data transfer of this register, when data transfer is active. These bits are DPSEL, MBSEL, DTDSEL, AC12EN, BCEN and DMAEN.

The Host Driver shall check the Command Inhibit DAT bit (CDIHB) and the Command Inhibit CMD bit (CIHB) in the Present State register before writing to this register. When the CDIHB bit in the Present State register is set, any attempt to send a command with data by writing to this register is ignored; when the CIHB bit is set, any write to this register is ignored.

On sending commands with data transfer invovled, it is mandatory that the block size is non-zero. Block count must also be non-zero, or indicated as single block transfer (bit 5 of this register is '0' when written), or block count is disabled (bit 1 of this register is '0' when written), otherwise uSDHC will ignore the sending of this command and do nothing. For write command, with all above restrictions, it is also mandatory that the write protect switch is not active (WPSPL bit of Present State Register is '1'), otherwise uSDHC will also ignore the command.

If the commands with data transfer does not receive the response in 64 clock cycles, i.e., response time-out, uSDHC will regard the external device does not accept the command and abort the data transfer. In this scenario, the driver should issue the command again to re-try the transfer. It is also possible that for some reason the card responds the command but uSDHC does not receive the response, and if it is internal DMA (either simple DMA or ADMA) read operation, the external system memory is over-written by the internal DMA with data sent back from the card.

The table below shows the summary of how register settings determine the type of data transfer.

Table 67-51. Transfer Type Register Setting for Various Transfer Types

Multi/Single Block Select	Block Count Enable	Block Count	Function
0	Don't Care	Don't Care	Single Transfer
1	0	Don't Care	Infinite Transfer
1	1	Positive Number	Multiple Transfer
1	1	Zero	No Data Transfer

The table below shows the relationship between the Command Index Check Enable and the Command CRC Check Enable, in regards to the Response Type bits as well as the name of the response type.

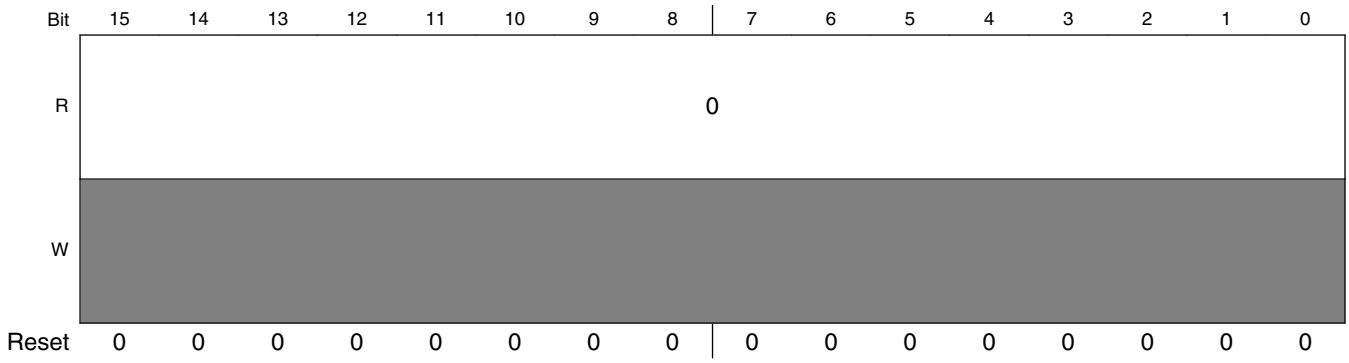
Table 67-52. Relationship Between Parameters and the Name of the Response Type

Response Type	Index Check Enable	CRC Check Enable	Name of Response Type
00	0	0	No Response
01	0	1	R2
10	0	0	R3,R4
10	1	1	R1,R5,R6
11	1	1	R1b,R5b

- In the SDIO specification, response type notation for R5b is not defined. R5 includes R5b in the SDIO specification. But R5b is defined in this specification to specify that the uSDHC will check the busy status after receiving a response. For example, usually CMD52 is used with R5, but the I/O abort command shall be used with R5b.
- The CRC field for R3 and R4 is expected to be all 1 bits. The CRC check shall be disabled for these response types.

Address: Base address + Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0		CMDINX[5:0]						CMDTYP[1:0]		DPSEL	CICEN	CCCEN	Reserved	RSPTYP[1:0]	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



uSDHCx_CMD_XFR_TYP field descriptions

Field	Description
31–30 Reserved	This read-only field is reserved and always has the value 0.
29–24 CMDINX[5:0]	Command Index: These bits shall be set to the command number that is specified in bits 45-40 of the Command-Format in the SD Memory Card Physical Layer Specification and SDIO Card Specification.
23–22 CMDTYP[1:0]	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 uSDHC shall assume that the card bus has been released and that it is possible to issue the next command which uses the DATA line. Since the uSDHC does not monitor the content of command response, it does not know if the Suspend command succeeded or not. It is the Host Driver's responsibility to check the status of the Suspend command and send another command marked as Suspend to inform the uSDHC that a Suspend command was successfully issued. Refer to Suspend Resume for more details. After the end bit of command is sent, the uSDHC de-asserts Read Wait for read transactions and stops checking busy for write transactions. In 4-bit mode, the interrupt cycle starts. If the Suspend command fails, the uSDHC will maintain its current state, and the Host Driver shall restart the transfer by setting the Continue Request bit in the Protocol Control register. • Resume Command: The Host Driver re-starts the data transfer by restoring the registers saved before sending the Suspend Command and then sends the Resume Command. The uSDHC will check for a pending busy state before starting write transfers. • Abort Command: If this command is set when executing a read transfer, the uSDHC will stop reads to the buffer. If this command is set when executing a write transfer, the uSDHC will stop driving the DATA line. After issuing the Abort command, the Host Driver should issue a software reset (Abort Transaction). 11 Abort CMD12, CMD52 for writing I/O Abort in CCCR 10 Resume CMD52 for writing Function Select in CCCR 01 Suspend CMD52 for writing Bus Suspend in CCCR 00 Normal Other commands
21 DPSEL	Data Present Select: This bit is set to 1 to indicate that data is present and shall be transferred using the DATA line. It is set to 0 for the following: • Commands using only the CMD line (e.g. CMD52). • Commands with no data transfer, but using the busy signal on DATA0 line (R1b or R5b e.g. CMD38)

Table continues on the next page...

uSDHCx_CMD_XFR_TYP field descriptions (continued)

Field	Description
	Note: In resume command, this bit shall be set, and other bits in this register shall be set the same as when the transfer was initially launched. When the Write Protect switch is on, (i.e. the WPSPL bit is active as '0'), any command with a write operation will be ignored. That is to say, when this bit is set, while the DTDSEL bit is 0, writes to the register Transfer Type are ignored. 1 Data Present 0 No Data Present
20 CICEN	Command Index Check Enable: If this bit is set to 1, the uSDHC will 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. 1 Enable 0 Disable
19 CCCEN	Command CRC Check Enable: If this bit is set to 1, the uSDHC 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 number of bits checked by the CRC field value changes according to the length of the response. (Refer to RSPTYP[1:0] and Command Transfer Type (uSDHC_CMD_XFR_TYP) .) 1 Enable 0 Disable
18 -	This field is reserved. Reserved
17–16 RSPTYP[1:0]	Response Type Select: 00 No Response 01 Response Length 136 10 Response Length 48 11 Response Length 48, check Busy after response
Reserved	This read-only field is reserved and always has the value 0.

67.8.5 Command Response0 (uSDHCx_CMD_RSP0)

This register is used to store part 0 of the response bits from the card.

Address: Base address + 10h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDRSP0[31:0]																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

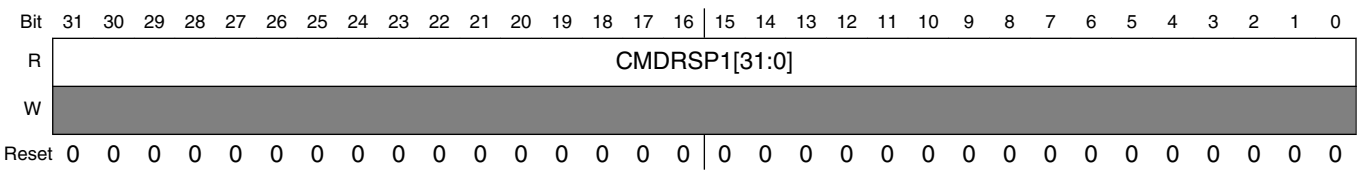
uSDHCx_CMD_RSP0 field descriptions

Field	Description
CMDRSP0[31:0]	Command Response 0: Refer to Command Response3 (uSDHC_CMD_RSP3) for the mapping of command responses from the SD Bus to this register for each response type.

67.8.6 Command Response1 (uSDHCx_CMD_RSP1)

This register is used to store part 1 of the response bits from the card.

Address: Base address + 14h offset



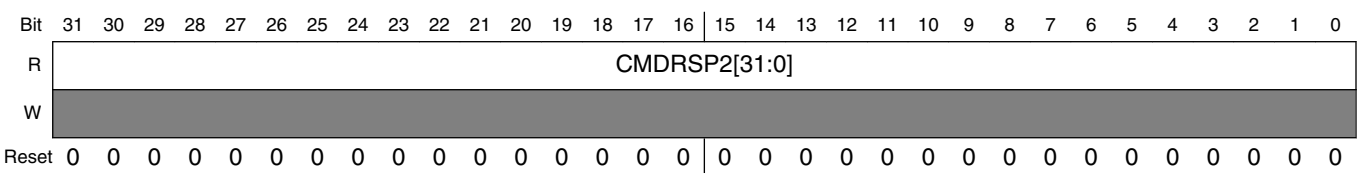
uSDHCx_CMD_RSP1 field descriptions

Field	Description
CMDRSP1[31:0]	Command Response 1: Refer to Command Response3 (uSDHC_CMD_RSP3) for the mapping of command responses from the SD Bus to this register for each response type.

67.8.7 Command Response2 (uSDHCx_CMD_RSP2)

This register is used to store part 2 of the response bits from the card.

Address: Base address + 18h offset



uSDHCx_CMD_RSP2 field descriptions

Field	Description
CMDRSP2[31:0]	Command Response 2: Refer to Command Response3 (uSDHC_CMD_RSP3) for the mapping of command responses from the SD Bus to this register for each response type.

67.8.8 Command Response3 (uSDHCx_CMD_RSP3)

This register is used to store part 3 of the response bits from the card.

The table below describes the mapping of command responses from the SD Bus to Command Response registers for each response type. In the table, R[ ] refers to a bit range within the response data as transmitted on the SD Bus.

Table 67-57. Response Bit Definition for Each Response Type

Response Type	Meaning of Response	Response Field	Response Register
R1,R1b (normal response)	Card Status	R[39:8]	CMDRSP0
R1b (Auto CMD12 response)	Card Status for Auto CMD12	R[39:8]	CMDRSP3
R2 (CID, CSD register)	CID/CSD register [127:8]	R[127:8]	{CMDRSP3[23:0], CMDRSP2, CMDRSP1, CMDRSP0}
R3 (OCR register)	OCR register for memory	R[39:8]	CMDRSP0
R4 (OCR register)	OCR register for I/O etc.	R[39:8]	CMDRSP0
R5, R5b	SDIO response	R[39:8]	CMDRSP0
R6 (Publish RCA)	New Published RCA[31:16] and card status[15:0]	R[39:9]	CMDRSP0

This table shows that most responses with a length of 48 (R[47:0]) have 32-bits of the response data (R[39:8]) stored in the CMDRSP0 register. Responses of type R1b (Auto CMD12 responses) have response data bits (R[39:8]) stored in the CMDRSP3 register. Responses with length 136 (R[135:0]) have 120-bits of the response data (R[127:8]) stored in the CMDRSP0, 1, 2, and 3 registers.

To be able to read the response status efficiently, the uSDHC only stores part of the response data in the Command Response registers. This enables the Host Driver to efficiently read 32-bits of response data in one read cycle on a 32-bit bus system. Parts of the response, the Index field and the CRC, are checked by the uSDHC (as specified by the Command Index Check Enable and the Command CRC Check Enable bits in the Transfer Type register) and generate an error interrupt if any error is detected. The bit range for the CRC check depends on the response length. If the response length is 48, the uSDHC will check R[47:1], and if the response length is 136 the uSDHC will check R[119:1].

Since the uSDHC may have a multiple block data transfer executing concurrently with a CMD_wo_DAT command, the uSDHC stores the Auto CMD12 response in the CMDRSP3 register. The CMD_wo_DAT response is stored in CMDRSP0. This allows

the uSDHC to avoid overwriting the Auto CMD12 response with the CMD_wo_DAT and vice versa. When the uSDHC modifies part of the Command Response registers, as shown in the table above, it preserves the unmodified bits.

Address: Base address + 1Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDRSP3[31:0]																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

uSDHCx_CMD_RSP3 field descriptions

Field	Description
CMDRSP3[31:0]	Command Response 3: Refer to Command Response3 (uSDHC_CMD_RSP3) for the mapping of command responses from the SD Bus to this register for each response type.

67.8.9 Data Buffer Access Port (uSDHCx_DATA_BUFF_ACC_PORT)

This is a 32-bit data port register used to access the internal buffer.

Address: Base address + 20h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATCONT[31:0]																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

uSDHCx_DATA_BUFF_ACC_PORT field descriptions

Field	Description
DATCONT[31:0]	Data Content: The Buffer Data Port register is for 32-bit data access by the ARM platform or the external DMA. When the internal DMA is enabled, any write to this register is ignored, and any read from this register will always yield 0s.

67.8.10 Present State (uSDHCx_PRES_STATE)

The Host Driver can get status of the uSDHC from this 32-bit read only register.

- The Host Driver can issue CMD0, CMD12, CMD13 (for memory) and CMD52 (for SDIO) when the DATA lines are busy during a data transfer. These commands can be issued when Command Inhibit (CMD) is set to zero. Other commands shall be

issued when Command Inhibit (DATA) is set to zero. Possible changes to the SD Physical Specification may add other commands to this list in the future.

- Note: the reset value of Present State Register depend on testbench connectivity.

Address: Base address + 24h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DLSL[7:0]								CLSL	0			WPSPL	CDPL	0	CINST
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			RTR	BREN	BWEN	RTA	WTA	SDOFF	PEROFF	HCKOFF	IPGOFF	SDSTB	DLA	CDIHB	CIHB
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

uSDHCx_PRES_STATE field descriptions

Field	Description
31–24 DLSL[7:0]	DATA[7:0] Line Signal Level: This status is used to check the DATA line level to recover from errors, and for debugging.This is especially useful in detecting the busy signal level from DATA0. The reset value is affected by the external pull-up/pull-down resistors. By default, the read value of this bit field after reset is 8'b11110111, when DATA3 is pulled down and the other lines are pulled up. DATA7: Data 7 line signal level DATA6: Data 6 line signal level DATA5: Data 5 line signal level DATA4: Data 4 line signal level DATA3: Data 3 line signal level

Table continues on the next page...

uSDHCx_PRES_STATE field descriptions (continued)

Field	Description
	DATA2: Data 2 line signal level DATA1: Data 1 line signal level DATA0: Data 0 line signal level
23 CLSL	CMD Line Signal Level: This status is used to check the CMD line level to recover from errors, and for debugging. The reset value is affected by the external pull-up/pull-down resistor, by default, the read value of this bit after reset is 1'b1, when the command line is pulled up.
22–20 Reserved	This read-only field is reserved and always has the value 0.
19 WPSPL	Write Protect Switch Pin Level: The Write Protect Switch is supported for memory and combo cards.This bit reflects the inverted value of the WP pin of the card socket. A software reset does not affect this bit. The reset value is effected by the external write protect switch. If the WP pin is not used, it should be tied low, so that the reset value of this bit is high and write is enabled. 1 Write enabled (WP=0) 0 Write protected (WP=1)
18 CDPL	Card Detect Pin Level: This bit reflects the inverse value of the CD_B pin for the card socket. Debouncing is not performed on this bit. This bit may be valid, but is not guaranteed, because of propagation delay. Use of this bit is limited to testing since it must be debounced by software. A software reset does not effect this bit. A write to the Force Event Register does not effect this bit. The reset value is effected by the external card detection pin. This bit shows the value on the CD_B pin (i.e. when a card is inserted in the socket, it is 0 on the CD_B input, and consequently the CDPL reads 1.) 1 Card present (CD_B=0) 0 No card present (CD_B=1)
17 Reserved	This read-only field is reserved and always has the value 0.
16 CINST	Card Inserted: This bit indicates whether a card has been inserted. The uSDHC debounces this signal so that the Host Driver will not need to wait for it to stabilize. Changing from a 0 to 1 generates a Card Insertion interrupt in the Interrupt Status register. Changing from a 1 to 0 generates a Card Removal interrupt in the Interrupt Status register. A write to the Force Event Register does not effect this bit. The Software Reset For All in the System Control register does not effect this bit.A software reset does not effect this bit. 1 Card Inserted 0 Power on Reset or No Card
15–13 Reserved	This read-only field is reserved and always has the value 0.
12 RTR	Re-Tuning Request: (only for SD3.0 SDR104 mode) Host Controller may request Host Driver to execute re-tuning sequence by setting this bit when the data window is shifted by temperature drift and a tuned sampling point does not have a good margin to receive correct data. This bit is cleared when a command is issued with setting Execute Tuning bit in MIXER_CTRL register.

Table continues on the next page...

uSDHCx_PRES_STATE field descriptions (continued)

Field	Description
	Changing of this bit from 0 to 1 generates Re-Tuning Event. Refer to Interrupt status registers for more detail. This bit isn't set to 1 if Sampling Clock Select in the MIXER_CTRL register is set to 0 (using fixed sampling clock). 1 Sampling clock needs re-tuning 0 Fixed or well tuned sampling clock
11 BREN	Buffer Read Enable: This status bit is used for non-DMA read transfers. The uSDHC implements an internal buffer to transfer data efficiently. This read only flag indicates that valid data exists in the host side buffer. If this bit is high, valid data greater than the watermark level exist in the buffer. A change of this bit from 1 to 0 occurs when some reads from the buffer(read DATPORT(Base + 0x20)) are made and the buffer hasn't valid data greater than the watermark level. A change of this bit from 0 to1 occurs when there is enough valid data ready in the buffer and the Buffer Read Ready interrupt has been generated and enabled. 1 Read enable 0 Read disable
10 BWEN	Buffer Write Enable: This status bit is used for non-DMA write transfers. The uSDHC implements an internal buffer to transfer data efficiently. This read only flag indicates if space is available for write data. If this bit is 1, valid data greater than the watermark level can be written to the buffer. A change of this bit from 1 to 0 occurs when some writes to the buffer(write DATPORT(Base + 0x20)) are made and the buffer hasn't valid space greater than the watermark level. . A change of this bit from 0 to 1 occurs when the buffer can hold valid data greater than the write watermark level and the Buffer Write Ready interrupt is generated and enabled. 1 Write enable 0 Write disable
9 RTA	Read Transfer Active: This status bit is used for detecting completion of a read transfer. This bit is set for either of the following conditions: • After the end bit of the read command. • When writing a 1 to the Continue Request bit in the Protocol Control register to restart a read transfer. A Transfer Complete interrupt is generated when this bit changes to 0. This bit is cleared for either of the following conditions: • When the last data block as specified by block length is transferred to the System, i.e. all data are read away from uSDHC internal buffer. • When all valid data blocks have been transferred from uSDHC internal buffer to the System and no current block transfers are being sent as a result of the Stop At Block Gap Request being set to 1. 1 Transferring data 0 No valid data
8 WTA	Write Transfer Active: This status bit indicates a write transfer is active. If this bit is 0, it means no valid write data exists in the uSDHC. This bit is set in either of the following cases:

Table continues on the next page...

uSDHCx_PRES_STATE field descriptions (continued)

Field	Description
	• After the end bit of the write command. • When writing 1 to the Continue Request bit in the Protocol Control register to restart a write transfer. This bit is cleared in either of the following cases: • After getting the CRC status of the last data block as specified by the transfer count (Single and Multiple). • After getting the CRC status of any block where data transmission is about to be stopped by a Stop At Block Gap Request. During a write transaction, a Block Gap Event interrupt is generated when this bit is changed to 0, as result of the Stop At Block Gap Request being set. This status is useful for the Host Driver in determining when to issue commands during Write Busy state. 1 Transferring data 0 No valid data
7 SDOFF	SD Clock Gated Off Internally: This status bit indicates that the SD Clock is internally gated off, because of buffer over/under-run or read pause without read wait assertion, or the driver set FRC_SDCLK_ON bit is 0 to stop the SD clock in idle status. Set IPG_PERCLK_SOFT_EN and CARD_CLK_SOFT_EN to 0 also gate off SD clock. This bit is for the Host Driver to debug data transaction on the SD bus. 1 SD Clock is gated off 0 SD Clock is active
6 PEROFF	ipg_perclk Gated Off Internally: This status bit indicates that the ipg_perclk is internally gated off. This bit is for the Host Driver to debug transaction on the SD bus. When IPG_CLK_SOFT_EN is cleared, ipg_perclk will be gated off, otherwise ipg_perclk will be always active. 1 ipg_perclk is gated off 0 ipg_perclk is active
5 HCKOFF	hclk Gated Off Internally: This status bit indicates that the hclk is internally gated off. This bit is for the Host Driver to debug during a data transfer. 1 hclk is gated off 0 hclk is active
4 IPGOFF	ipg_clk Gated Off Internally: This status bit indicates that the ipg_clk is internally gated off. This bit is for the Host Driver to debug. 1 ipg_clk is gated off 0 ipg_clk is active
3 SDSTB	SD Clock Stable This status bit indicates that the internal card clock is stable. This bit is for the Host Driver to poll clock status when changing the clock frequency. It is recommended to clear FRC_SDCLK_ON bit in System Control register to remove glitches on the card clock when the frequency is changing. *Before changing clock divisor value(SDCLKFS or DVS), Host Driver should make sure the SDSTB bit is high.*

Table continues on the next page...

uSDHCx_PRES_STATE field descriptions (continued)

Field	Description
	1 clock is stable 0 clock is changing frequency and not stable
2 DLA	Data Line Active This status bit indicates whether one of the DATA lines on the SD Bus is in use. In the case of read transactions: This status indicates if a read transfer is executing on the SD Bus. Changes in this value from 1 to 0, between data blocks, generates a Block Gap Event interrupt in the Interrupt Status register. This bit will be set in either of the following cases: • After the end bit of the read command. • When writing a 1 to the Continue Request bit in the Protocol Control register to restart a read transfer. This bit will be cleared in either of the following cases: (1) When the end bit of the last data block is sent from the SD Bus to the uSDHC. (2) When the Read Wait state is stopped by a Suspend command and the DATA2 line is released. The uSDHC will wait at the next block gap by driving Read Wait at the start of the interrupt cycle. If the Read Wait signal is already driven (data buffer cannot receive data), the uSDHC can wait for a current block gap by continuing to drive the Read Wait signal. It is necessary to support Read Wait in order to use the suspend / resume function. This bit will remain 1 during Read Wait. In the case of write transactions: This status indicates that a write transfer is executing on the SD Bus. Changes in this value from 1 to 0 generate a Transfer Complete interrupt in the Interrupt Status register. This bit will be set in either of the following cases: • After the end bit of the write command. • When writing to 1 to the Continue Request bit in the Protocol Control register to continue a write transfer. This bit will be cleared in either of the following cases: • When the SD card releases Write Busy of the last data block, the uSDHC will also detect if the output is not busy. If the SD card does not drive the busy signal after the CRC status is received, the uSDHC shall assume the card drive "Not Busy". • When the SD card releases write busy, prior to waiting for write transfer, and as a result of a Stop At Block Gap Request. In the case of command with busy pending: This status indicates that a busy state follows the command and the data line is in use. This bit will be cleared when the DATA0 line is released. 1 DATA Line Active 0 DATA Line Inactive
1 CDIHB	Command Inhibit (DATA): This status bit is generated if either the DAT Line Active or the Read Transfer Active is set to 1. If this bit is 0, it indicates that the uSDHC can issue the next SD/MMC Command. Commands with a busy signal belong to Command Inhibit (DATA) (for example. R1b, R5b type). Changing from 1 to 0 generates a Transfer Complete interrupt in the Interrupt Status register. Note: The SD Host Driver can save registers for a suspend transaction after this bit has changed from 1 to 0.

Table continues on the next page...

uSDHCx_PRES_STATE field descriptions (continued)

Field	Description
	1 Cannot issue command which uses the DATA line 0 Can issue command which uses the DATA line
0 CIHB	Command Inhibit (CMD): If this status bit is 0, it indicates that the CMD line is not in use and the uSDHC can issue a SD/MMC Command using the CMD line. This bit is set also immediately after the Transfer Type register is written. This bit is cleared when the command response is received. Even if the Command Inhibit (DATA) is set to 1, Commands using only the CMD line can be issued if this bit is 0. Changing from 1 to 0 generates a Command Complete interrupt in the Interrupt Status register. If the uSDHC cannot issue the command because of a command conflict error (Refer to Command CRC Error) or because of a Command Not Issued By Auto CMD12 Error, this bit will remain 1 and the Command Complete is not set. The Status of issuing an Auto CMD12 does not show on this bit. 1 Cannot issue command 0 Can issue command using only CMD line

67.8.11 Protocol Control (uSDHCx_PROT_CTRL)

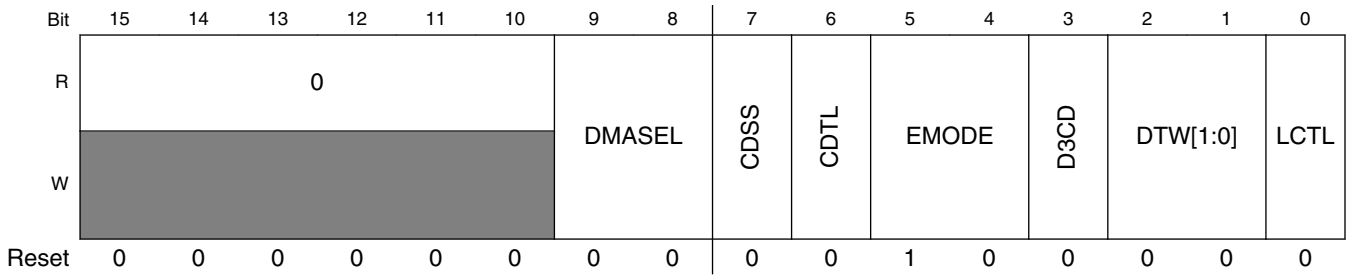
There are three cases to restart the transfer after stop at the block gap. Which case is appropriate depends on whether the uSDHC issues a Suspend command or the SD card accepts the Suspend command.

1. If the Host Driver does not issue a Suspend command, the Continue Request shall be used to restart the transfer.
2. If the Host Driver issues a Suspend command and the SD card accepts it, a Resume command shall be used to restart the transfer.
3. If the Host Driver issues a Suspend command and the SD card does not accept it, the Continue Request shall be used to restart the transfer.

Any time Stop At Block Gap Request stops the data transfer, the Host Driver shall wait for a Transfer Complete (in the Interrupt Status register), before attempting to restart the transfer. When restarting the data transfer by Continue Request, the Host Driver shall clear the Stop At Block Gap Request before or simultaneously.

Address: Base address + 28h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-	NON_EXACT_BLK_RD	BURST_LEN_EN			WECRM	WECINS	WECINT	-			RD_DONE_NO_8CLK	IABG	RWCTL	CREQ	SABGREQ
Reset	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0



uSDHCx_PROT_CTRL field descriptions

Field	Description
31 -	Reserved. Always write as 0
30 NON_EXACT_ BLK_RD	Current block read is non-exact block read. It's only used for SDIO. 1 The block read is non-exact block read. Host driver needs to issue abort command to terminate this multi-block read. 0 The block read is exact block read. Host driver doesn't need to issue abort command to terminate this multi-block read.
29–27 BURST_LEN_EN	BURST length enable for INCR, INCR4/INCR8/INCR16, INCR4-WRAP/INCR8-WRAP/INCR16-WRAP This is used to enable/disable the burst length for the external AHB2AXI bridge. It's useful especially for INCR transfer because without burst length indicator, the AHB2AXI bridge doesn't know the burst length in advance. Without burst length indicator, AHB INCR transfers can only be converted to SINGLEs on the AXI side. xx1 Burst length is enabled for INCR x1x Burst length is enabled for INCR4/INCR8/INCR16 1xx Burst length is enabled for INCR4-WRAP/INCR8-WRAP/INCR16-WRAP
26 WECRM	Wakeup Event Enable On SD Card Removal: This bit enables a wakeup event, via a Card Removal, in the Interrupt Status register. FN_WUS (Wake Up Support) in CIS does not effect this bit. When this bit is set, the Card Removal Status and the uSDHC interrupt can be asserted without CLK toggling. When the wakeup feature is not enabled, the CLK must be active in order to assert the Card Removal Status and the uSDHC interrupt. 1 Enable 0 Disable
25 WECINS	Wakeup Event Enable On SD Card Insertion: This bit enables a wakeup event, via a Card Insertion, in the Interrupt Status register. FN_WUS (Wake Up Support) in CIS does not effect this bit. When this bit is set, the Card Insertion Status and the uSDHC interrupt can be asserted without CLK toggling. When the wakeup feature is not enabled, the CLK must be active in order to assert the Card Insertion Status and the uSDHC interrupt. 1 Enable 0 Disable
24 WECINT	Wakeup Event Enable On Card Interrupt: This bit enables a wakeup event, via a Card Interrupt, in the Interrupt Status register. This bit can be set to 1 if FN_WUS (Wake Up Support) in CIS is set to 1. When this bit is set, the Card Interrupt Status and the uSDHC interrupt can be asserted without CLK toggling. When the wakeup feature is not enabled, the CLK must be active in order to assert the Card Interrupt Status and the uSDHC interrupt.

Table continues on the next page...

uSDHCx_PROT_CTRL field descriptions (continued)

Field	Description
	1 Enable 0 Disable
23–21 -	Reserved. Always write as 3'b100
20 RD_DONE_NO_8CLK	*Read done no 8 clock:* *According to the SD/MMC spec, for read data transaction, 8 clocks are needed after the end bit of the last data block. So, by default(RD_DONE_NO_8CLK=0), 8 clocks will be active after the end bit of the last read data transaction.* *However, this 8 clocks should not be active if user wants to use stop at block gap(include the auto stop at block gap in boot mode) feature for read and the RWCTL bit(bit18) is not enabled. In this case, software should set RD_DONE_NO_8CLK to avoid this 8 clocks. Otherwise, the device may send extra data to uSDHC while uSDHC ignores these data.* *In a summary, this bit should be set only if the use case needs to use stop at block gap feature while the device can't support the read wait feature.*
19 IABG	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 SDIO card can't signal an interrupt during a multiple block transfer, this bit should be set to 0 to avoid an inadvertent interrupt. When the Host Driver detects an SDIO card insertion, it shall set this bit according to the CCCR of the card. 1 Enabled 0 Disabled
18 RWCTL	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 DATA2 line. Otherwise the uSDHC has to stop the SD Clock to hold read data, which restricts commands generation. When the Host Driver detects an SDIO card insertion, it shall set this bit according to the CCCR of the card. If the card does not support read wait, this bit shall never be set to 1, otherwise DATA line conflicts may occur. If this bit is set to 0, stop at block gap during read operation is also supported, but the uSDHC will stop the SD Clock to pause reading operation. 1 Enable Read Wait Control, and assert Read Wait without stopping SD Clock at block gap when SABGREQ bit is set 0 Disable Read Wait Control, and stop SD Clock at block gap when SABGREQ bit is set
17 CREQ	Continue Request: This bit is used to restart a transaction which was stopped using the Stop At Block Gap Request. When a Suspend operation is not accepted by the card, it is also by setting this bit to restart the paused transfer. To cancel stop at the block gap, set Stop At Block Gap Request to 0 and set this bit to 1 to restart the transfer. The uSDHC automatically clears this bit, therefore it is not necessary for the Host Driver to set this bit to 0. If both Stop At Block Gap Request and this bit are 1, the continue request is ignored. 1 Restart 0 No effect
16 SABGREQ	Stop At Block Gap Request:

Table continues on the next page...

uSDHCx_PROT_CTRL field descriptions (continued)

Field	Description
	This bit is used to stop executing a transaction at the next block gap for both DMA and non-DMA transfers. Until the Transfer Complete is set to 1, indicating a transfer completion, the Host Driver shall leave this bit set to 1. Clearing both the Stop At Block Gap Request and Continue Request does not cause the transaction to restart. Read Wait is used to stop the read transaction at the block gap. The uSDHC will honor the Stop At Block Gap Request for write transfers, but for read transfers it requires that the SDIO card support Read Wait. Therefore, the Host Driver shall not set this bit during read transfers unless the SDIO card supports Read Wait and has set the Read Wait Control to 1, otherwise the uSDHC will stop the SD bus clock to pause the read operation during block gap. In the case of write transfers in which the Host Driver writes data to the 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 the Data Port register after a block is sent. Once this bit is set, the Host Driver shall not clear this bit before the Transfer Complete bit in Interrupt Status Register is set, otherwise the uSDHCs behavior is undefined. This bit effects Read Transfer Active, Write Transfer Active, DATA Line Active and Command Inhibit (DATA) in the Present State register. 1 Stop 0 Transfer
15–10 Reserved	This read-only field is reserved and always has the value 0.
9–8 DMASEL	DMA Select: This field is valid while DMA (SDMA or ADMA) is enabled and selects the DMA operation. 00 No DMA or Simple DMA is selected 01 ADMA1 is selected 10 ADMA2 is selected 11 reserved
7 CDSS	Card Detect Signal Selection: This bit selects the source for the card detection. 1 Card Detection Test Level is selected (for test purpose) 0 Card Detection Level is selected (for normal purpose)
6 CDTL	Card Detect Test Level: This is bit is enabled while the Card Detection Signal Selection is set to 1 and it indicates card insertion. 1 Card Detect Test Level is 1, card inserted 0 Card Detect Test Level is 0, no card inserted
5–4 EMODE	Endian Mode: The uSDHC supports all three endian modes in data transfer. Refer to Data Buffer for more details. 00 Big Endian Mode 01 Half Word Big Endian Mode 10 Little Endian Mode 11 Reserved
3 D3CD	DATA3 as Card Detection Pin: If this bit is set, DATA3 should be pulled down to act as a card detection pin. Be cautious when using this feature, because DATA3 is also a chip-select for the SPI mode. A pull-down on this pin and CMD0 may set the card into the SPI mode, which the uSDHC does not support.

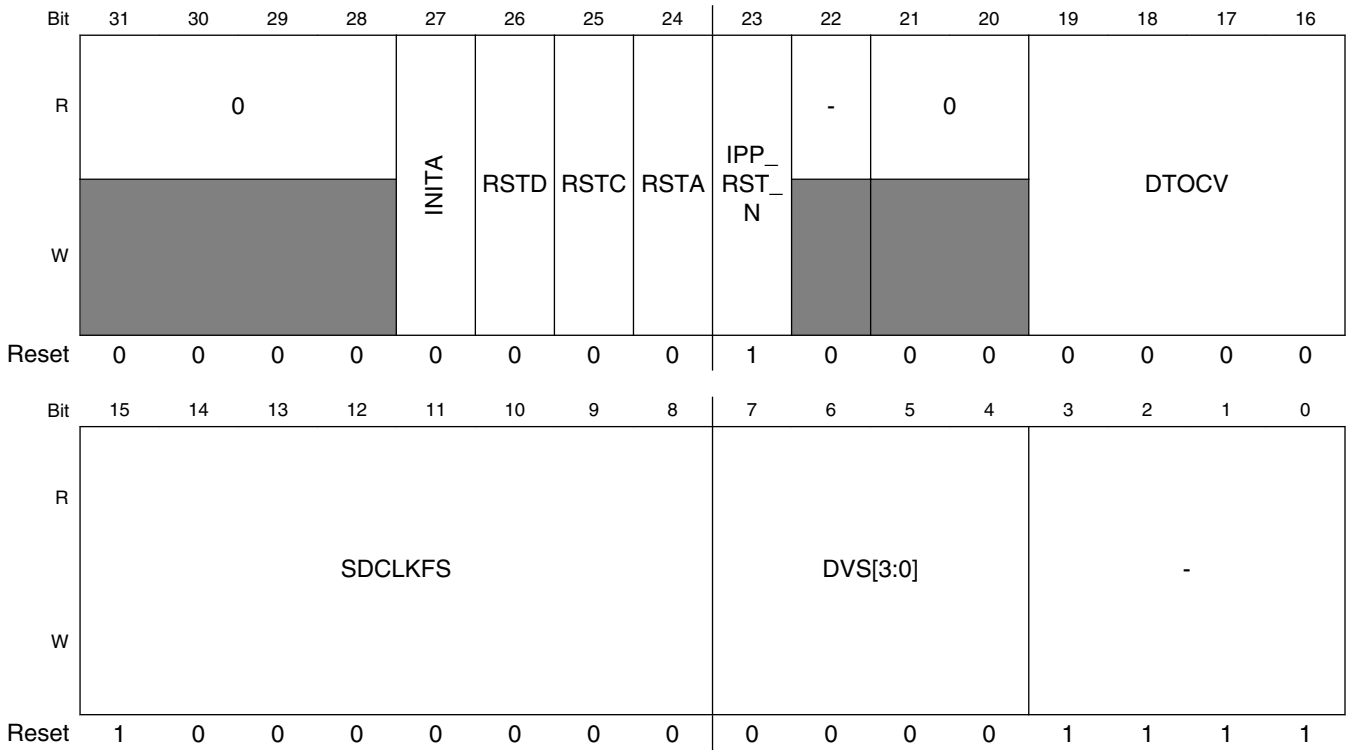
Table continues on the next page...

uSDHCx_PROT_CTRL field descriptions (continued)

Field	Description
	1 DATA3 as Card Detection Pin 0 DATA3 does not monitor Card Insertion
2–1 DTW[1:0]	Data Transfer Width: This bit selects the data width of the SD bus for a data transfer. The Host Driver shall set it to match the data width of the card. Possible Data transfer Width is 1-bit, 4-bits or 8-bits. 10 8-bit mode 01 4-bit mode 00 1-bit mode 11 Reserved
0 LCTL	LED Control: This bit, fully controlled by the Host Driver, is used to caution the user not to remove the card while the 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. When the software issues multiple SD commands, setting the bit once before the first command is sufficient: it is not necessary to reset the bit between commands. 1 LED on 0 LED off

67.8.12 System Control (uSDHCx_SYS_CTRL)

Address: Base address + 2Ch offset



uSDHCx_SYS_CTRL field descriptions

Field	Description
31–28 Reserved	This read-only field is reserved and always has the value 0.
27 INITA	Initialization Active: When this bit is set, 80 SD-Clocks are sent to the card. After the 80 clocks are sent, this bit is self cleared. This bit is very useful during the card power-up period when 74 SD-Clocks are needed and the clock auto gating feature is enabled. Writing 1 to this bit when this bit is already 1 has no effect. Writing 0 to this bit at any time has no effect. When either of the CIHB and CDIHB bits in the Present State Register are set, writing 1 to this bit is ignored (i.e. when command line or data lines are active, write to this bit is not allowed). On the otherhand, when this bit is set, i.e., during intialization active period, it is allowed to issue command, and the command bit stream will appear on the CMD pad after all 80 clock cycles are done. So when this command ends, the driver can make sure the 80 clock cycles are sent out. This is very useful when the driver needs send 80 cycles to the card and does not want to wait till this bit is self cleared.
26 RSTD	Software Reset For DATA Line: Only part of the data circuit is reset. DMA circuit is also reset. After this bit is set, SW waits for self-clear. The following registers and bits are cleared by this bit: • Data Port register

Table continues on the next page...

uSDHCx_SYS_CTRL field descriptions (continued)

Field	Description
	• Buffer is cleared and initialized. • Present State register • Buffer Read Enable • Buffer Write Enable • Read Transfer Active • Write Transfer Active • DATA Line Active • Command Inhibit (DATA) Protocol Control register • Continue Request • Stop At Block Gap Request Interrupt Status register • Buffer Read Ready • Buffer Write Ready • DMA Interrupt • Block Gap Event • Transfer Complete 1 Reset 0 No Reset
25 RSTC	Software Reset For CMD Line: Only part of the command circuit is reset. After this bit is set, SW waits for self-clear. The following registers and bits are cleared by this bit: • Present State register Command Inhibit (CMD) • Interrupt Status register Command Complete 1 Reset 0 No Reset
24 RSTA	Software Reset For ALL: This reset effects the entire Host Controller except for the card detection circuit. Register bits of type ROC, RW, RW1C, RWAC are cleared. During its initialization, the Host Driver shall set this bit to 1 to reset the uSDHC. The uSDHC shall reset this bit to 0 when the capabilities registers are valid and the Host Driver can read them. Additional use of Software Reset For All does not affect the value of the Capabilities registers. After this bit is set, it is recommended that the Host Driver reset the external card and re-initialize it. After this bit is set, SW should wait for self-clear. 1 Reset 0 No Reset
23 IPP_RST_N	This register's value will be output to CARD from pad directly for hardware reset of the card if the card supports this feature.
22 -	Reserved
21–20 Reserved	This read-only field is reserved and always has the value 0.
19–16 DTOCV	Data Timeout Counter Value: This value determines the interval by which DAT line timeouts are detected. Refer to the Data Timeout Error bit in the Interrupt Status register for information on factors that dictate time-out generation. Time-out clock frequency will be generated by dividing the base clock SDCLK value by this value. The Host Driver can clear the Data Timeout Error Status Enable (in the Interrupt Status Enable register) to prevent inadvertent time-out events. 1111 SDCLK x 2^{28}

Table continues on the next page...

uSDHCx_SYS_CTRL field descriptions (continued)

Field	Description
	1110 SDCLK x 2 ^{27} 0001 SDCLK x 2 ^{14} 0000 SDCLK x 2 ^{13}
15–8 SDCLKFS	SDCLK Frequency Select: This register is used to select the frequency of the SDCLK pin. The frequency is not programmed directly, rather this register holds the prescaler (this register) and divisor (next register) of the Base Clock Frequency register. *In Single Data Rate mode(DDR_EN bit of MIXERCTRL is '0')* Only the following settings are allowed: 80h) Base clock divided by 256 40h) Base clock divided by 128 20h) Base clock divided by 64 10h) Base clock divided by 32 08h) Base clock divided by 16 04h) Base clock divided by 8 02h) Base clock divided by 4 01h) Base clock divided by 2 00h) Base clock divided by 1 *While in Dual Data Rate mode(DDR_EN bit of MIXERCTRL is '1')* *Only the following settings are allowed:* *80h) Base clock divided by 512* *40h) Base clock divided by 256* *20h) Base clock divided by 128* *10h) Base clock divided by 64* *08h) Base clock divided by 32* *04h) Base clock divided by 16* *02h) Base clock divided by 8* *01h) Base clock divided by 4* *00h) Base clock divided by 2* *When S/W changes the DDR_EN bit, SDCLKFS may need to be changed also!* In Single Data Rate mode, setting 00h bypasses the frequency prescaler of the SD Clock. Multiple bits must not be set, or the behavior of this prescaler is undefined. The two default divider values can be calculated by the frequency of ipg_perclk and the following Divisor bits. The frequency of SDCLK is set by the following formula: Clock Frequency = (Base Clock) / (prescaler x divisor) For example, in Single Data Rate mode, if the Base Clock Frequency is 96 MHz, and the target frequency is 25 MHz, then choosing the prescaler value of 01h and divisor value of 1h will yield 24 MHz, which is the nearest frequency less than or equal to the target. Similarly, to approach a clock value of 400 kHz, the prescaler value of 08h and divisor value of eh yields the exact clock value of 400 kHz.

Table continues on the next page...

uSDHCx_SYS_CTRL field descriptions (continued)

Field	Description
	The reset value of this bit field is 80h, so if the input Base Clock (ipg_perclk) is about 96 MHz, the default SD Clock after reset is 375 kHz. According to the SD Physical Specification Version 1.1 and the SDIO Card Specification Version 1.2, the maximum SD Clock frequency is 50 MHz and shall never exceed this limit. *Before changing clock divisor value(SDCLKFS or DVS), Host Driver should make sure the SDSTB bit is high.* *If setting SDCLKFS and DVS can generate same clock frequency,(For example, in SDR mode, SDCLKFS = 01h is same as DVS = 01h.) SDCLKFS is highly recommended.*
7–4 DVS[3:0]	Divisor: This register is used to provide a more exact divisor to generate the desired SD clock frequency. Note the divider can even support odd divisors without deterioration of duty cycle. *Before changing clock divisor value(SDCLKFS or DVS), Host Driver should make sure the SDSTB bit is high.* The setting are as following: 0000 Divide-by-1 0001 Divide-by-2 ..... 1110 Divide-by-15 1111 Divide-by-16
-	Reserved. Always write as 1.

67.8.13 Interrupt Status (uSDHCx_INT_STATUS)

An interrupt is generated when the Normal Interrupt Signal Enable is enabled and at least one of the status bits is set to 1. For all bits, writing 1 to a bit clears it; writing to 0 keeps the bit unchanged. More than one status can be cleared with a single register write. For Card Interrupt, before writing 1 to clear, it is required that the card stops asserting the interrupt, meaning that when the Card Driver services the interrupt condition, otherwise the CINT bit will be asserted again.

The table below shows the relationship between the Command Timeout Error and the Command Complete.

Table 67-63. uSDHC Status for Command Timeout Error/Command Complete Bit Combinations

Command Complete	Command Timeout Error	Meaning of the Status
0	0	X
X	1	Response not received within 64 SDCLK cycles
1	0	Response received

The table below shows the relationship between the Transfer Complete and the Data Timeout Error.

Table 67-64. uSDHC Status for Data Timeout Error/Transfer Complete Bit Combinations

Transfer Complete	Data Timeout Error	Meaning of the Status
0	0	X
0	1	Timeout occurred during transfer
1	X	Data Transfer Complete

The table below shows the relationship between the Command CRC Error and Command Timeout Error.

Table 67-65. uSDHC Status for Command CRC Error/Command Timeout Error Bit Combinations

Command Complete	Command Timeout Error	Meaning of the Status
0	0	No error
0	1	Response Timeout Error
1	0	Response CRC Error
1	1	CMD line conflict

Address: Base address + 30h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			DMAE	0	TNE	0	AC12E	0	DEBE	DCE	DTOE	CIE	CEBE	CCE	CTOE
W				w1c		w1c		w1c		w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	TP	0	RTE	0			CINT	CRM	CINS	BRR	BWR	DINT	BGE	TC	CC
W		w1c		w1c				w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

uSDHCx_INT_STATUS field descriptions

Field	Description
31–29 Reserved	This read-only field is reserved and always has the value 0.
28 DMAE	DMA Error: Occurs when an Internal DMA transfer has failed. This bit is set to 1, when some error occurs in the data transfer. This error can be caused by either Simple DMA or ADMA, depending on which DMA is in use. The value in DMA System Address register is the next fetch address where the error occurs. Since any error corrupts the whole data block, the Host Driver shall re-start the transfer from the corrupted block boundary. The address of the block boundary can be calculated either from the current DS_ADDR value or from the remaining number of blocks and the block size. 1 Error 0 No Error
27 Reserved	This read-only field is reserved and always has the value 0.
26 TNE	Tuning Error: (only for SD3.0 SDR104 mode) This bit is set when an unrecoverable error is detected in a tuning circuit. By detecting Tuning Error, Host Driver needs to abort a command executing and perform tuning.
25 Reserved	This read-only field is reserved and always has the value 0.
24 AC12E	Auto CMD12 Error: Occurs when detecting that one of the bits in the 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 the Auto CMD12 is not executed due to the previous command error. 1 Error 0 No Error
23 Reserved	This read-only field is reserved and always has the value 0.
22 DEBE	Data End Bit Error: Occurs either when detecting 0 at the end bit position of read data, which uses the DATA line, or at the end bit position of the CRC. 1 Error 0 No Error
21 DCE	Data CRC Error: Occurs when detecting a CRC error when transferring read data, which uses the DATA line, or when detecting the Write CRC status having a value other than 010. 1 Error 0 No Error
20 DTOE	Data Timeout Error: Occurs when detecting one of following time-out conditions. • Busy time-out for R1b,R5b type • Busy time-out after Write CRC status • Read Data time-out. 1 Time out 0 No Error

Table continues on the next page...

uSDHCx_INT_STATUS field descriptions (continued)

Field	Description
19 CIE	Command Index Error: Occurs if a Command Index error occurs in the command response. 1 Error 0 No Error
18 CEBE	Command End Bit Error: Occurs when detecting that the end bit of a command response is 0. 1 End Bit Error Generated 0 No Error
17 CCE	Command CRC Error: Command CRC Error is generated in two cases. • If a response is returned and the Command Timeout Error is set to 0 (indicating no time-out), this bit is set when detecting a CRC error in the command response. • The uSDHC detects a CMD line conflict by monitoring the CMD line when a command is issued. If the uSDHC drives the CMD line to 1, but detects 0 on the CMD line at the next SDCLK edge, then the uSDHC 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. 1 CRC Error Generated. 0 No Error
16 CTOE	Command Timeout Error: Occurs only if no response is returned within 64 SDCLK cycles from the end bit of the command. If the uSDHC detects a CMD line conflict, in which case a Command CRC Error shall also be set (as shown in Interrupt Status (uSDHC_INT_STATUS)), this bit shall be set without waiting for 64 SDCLK cycles. This is because the command will be aborted by the uSDHC. 1 Time out 0 No Error
15 Reserved	This read-only field is reserved and always has the value 0.
14 TP	Tuning Pass:(only for SD3.0 SDR104 mode) Current CMD19 transfer is done successfully. That is, current sampling point is correct.
13 Reserved	This read-only field is reserved and always has the value 0.
12 RTE	Re-Tuning Event: (only for SD3.0 SDR104 mode) This status is set if Re-Tuning Request in the Present State register changes from 0 to 1. Host Controller requests Host Driver to perform re-tuning for next data transfer. Current data transfer (not large block count) can be completed without re-tuning. 1 Re-Tuning should be performed 0 Re-Tuning is not required
11–9 Reserved	This read-only field is reserved and always has the value 0.
8 CINT	Card Interrupt:

Table continues on the next page...

uSDHCx_INT_STATUS field descriptions (continued)

Field	Description
	This status bit is set when an interrupt signal is detected from the external card. In 1-bit mode, the uSDHC will detect the Card Interrupt without the SD Clock to support wakeup. In 4-bit mode, the card interrupt signal is sampled during the interrupt cycle, so the interrupt from card can only be sampled during interrupt cycle, introducing some delay between the interrupt signal from the SDIO card and the interrupt to the Host System. Writing this bit to 1 can clear this bit, but as the interrupt source from the SDIO card does not clear, this bit is set again. In order to clear this bit, it is required to reset the interrupt source from the external card followed by a writing 1 to this bit. When this status has been set, and the Host Driver needs to service this interrupt, the Card Interrupt Signal Enable in the Interrupt Signal Enable register should be 0 to stop driving the interrupt signal to the Host System. After completion of the card interrupt service (It should reset the interrupt sources in the SDIO card and the interrupt signal may not be asserted), write 1 to clear this bit, set the Card Interrupt Signal Enable to 1, and start sampling the interrupt signal again. 1 Generate Card Interrupt 0 No Card Interrupt
7 CRM	Card Removal: This status bit is set if the Card Inserted bit 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 state may possibly be changed when the Host Driver clears this bit and the interrupt event may not be generated. When this bit is cleared, it will be set again if no card is inserted. In order to leave it cleared, clear the Card Removal Status Enable bit in Interrupt Status Enable register. 1 Card removed 0 Card state unstable or inserted
6 CINS	Card Insertion: This status bit is set if the Card Inserted bit 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 state may possibly be changed when the Host Driver clears this bit and the interrupt event may not be generated. When this bit is cleared, it will be set again if a card is inserted. In order to leave it cleared, clear the Card Inserted Status Enable bit in Interrupt Status Enable register. 1 Card inserted 0 Card state unstable or removed
5 BRR	Buffer Read Ready: This status bit is set if the Buffer Read Enable bit, in the Present State register, changes from 0 to 1. Refer to the Buffer Read Enable bit in the Present State register for additional information. 1 Ready to read buffer 0 Not ready to read buffer
4 BWR	Buffer Write Ready: This status bit is set if the Buffer Write Enable bit, in the Present State register, changes from 0 to 1. Refer to the Buffer Write Enable bit in the Present State register for additional information. 1 Ready to write buffer: 0 Not ready to write buffer
3 DINT	DMA Interrupt:

Table continues on the next page...

uSDHCx_INT_STATUS field descriptions (continued)

Field	Description
	Occurs only when the internal DMA finishes the data transfer successfully. Whenever errors occur during data transfer, this bit will not be set. Instead, the DMAE bit will be set. Either Simple DMA or ADMA finishes data transferring, this bit will be set. 1 DMA Interrupt is generated 0 No DMA Interrupt
2 BGE	Block Gap Event: If the Stop At Block Gap Request bit in the Protocol Control register is set, this bit is set when a read or 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. In the case of a Read Transaction: This bit is set at the falling edge of the DATA Line Active Status (When the transaction is stopped at SD Bus timing). The Read Wait must be supported in order to use this function. In the case of Write Transaction: This bit is set at the falling edge of Write Transfer Active Status (After getting CRC status at SD Bus timing). 1 Transaction stopped at block gap 0 No block gap event
1 TC	Transfer Complete: This bit is set when a read or write transfer is completed. In the case of a Read Transaction: This bit is set at the falling edge of the Read Transfer Active Status. There are two cases in which this interrupt is generated. The first is when a data transfer is completed as specified by the data length (after the last data has been read to the Host System). The second is when data has stopped at the block gap and completed the data transfer by setting the Stop At Block Gap Request bit in the Protocol Control register (after valid data has been read to the Host System). In the case of a Write Transaction: This bit is set at the falling edge of the DATA Line Active Status. There are two cases in which this interrupt is generated. The first is when the last data is written to the SD card as specified by the data length and the busy signal is released. The second is when data transfers are stopped at the block gap, by setting the Stop At Block Gap Request bit in the Protocol Control register, and the data transfers are completed. (after valid data is written to the SD card and the busy signal released). In the case of a command with busy, this bit is set when busy is deasserted. 1 Transfer complete 0 Transfer not complete
0 CC	Command Complete: This bit is set when you receive the end bit of the command response (except Auto CMD12). Refer to the Command Inhibit (CMD) in the Present State register. 1 Command complete 0 Command not complete

67.8.14 Interrupt Status Enable (uSDHCx_INT_STATUS_EN)

Setting the bits in this register to 1 enables the corresponding Interrupt Status to be set by the specified event. If any bit is cleared, the corresponding Interrupt Status bit is also cleared (i.e. when the bit in this register is cleared, the corresponding bit in Interrupt Status Register is always 0).

- Depending on IABG bit setting, uSDHC may be programmed to sample the card interrupt signal during the interrupt period and hold its value in the flip-flop. There will be some delays on the Card Interrupt, asserted from the card, to the time the Host System is informed.
- To detect a CMD line conflict, the Host Driver must set both Command Timeout Error Status Enable and Command CRC Error Status Enable to 1.

Address: Base address + 34h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			DMAESEN	0	TNESEN	0	AC12ESEN	0	DEBESEN	DCESEN	DTOESEN	CIESEN	CEBESEN	CCESEN	CTOESEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	TPSEN	0	RTESEN	0			CINTSEN	CRMSEN	CINSSEN	BRRSEN	BWRSEN	DINTSEN	BGESEN	TCSEN	CCSEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

uSDHCx_INT_STATUS_EN field descriptions

Field	Description
31–29 Reserved	This read-only field is reserved and always has the value 0.
28 DMAESEN	DMA Error Status Enable: 1 Enabled 0 Masked
27 Reserved	This read-only field is reserved and always has the value 0.
26 TNESEN	Tuning Error Status Enable: 1 Enabled 0 Masked
25 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

uSDHCx_INT_STATUS_EN field descriptions (continued)

Field	Description
24 AC12ESEN	Auto CMD12 Error Status Enable: 1 Enabled 0 Masked
23 Reserved	This read-only field is reserved and always has the value 0.
22 DEBESEN	Data End Bit Error Status Enable: 1 Enabled 0 Masked
21 DCESEN	Data CRC Error Status Enable: 1 Enabled 0 Masked
20 DTOESEN	Data Timeout Error Status Enable: 1 Enabled 0 Masked
19 CIESEN	Command Index Error Status Enable: 1 Enabled 0 Masked
18 CEBESEN	Command End Bit Error Status Enable: 1 Enabled 0 Masked
17 CCESEN	Command CRC Error Status Enable: 1 Enabled 0 Masked
16 CTOESEN	Command Timeout Error Status Enable: 1 Enabled 0 Masked
15 Reserved	This read-only field is reserved and always has the value 0.
14 TPSEN	Tuning Pass Status Enable 1 Enabled 0 Masked
13 Reserved	This read-only field is reserved and always has the value 0.
12 RTESEN	Re-Tuning Event Status Enable 1 Enabled 0 Masked
11–9 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

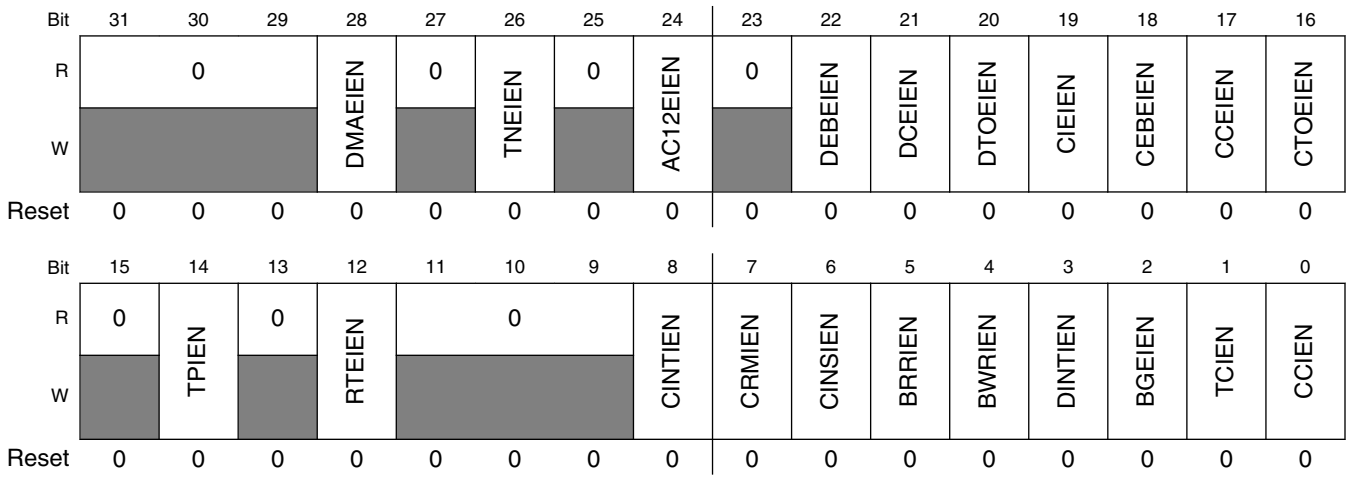
uSDHCx_INT_STATUS_EN field descriptions (continued)

Field	Description
8 CINTSEN	Card Interrupt Status Enable: If this bit is set to 0, the uSDHC will clear the interrupt request to the System. The Card Interrupt detection is stopped when this bit is cleared and restarted when this bit is set to 1. The Host Driver should clear the Card Interrupt Status Enable before servicing the Card Interrupt and should set this bit again after all interrupt requests from the card are cleared to prevent inadvertent interrupts. 1 Enabled 0 Masked
7 CRMSEN	Card Removal Status Enable: 1 Enabled 0 Masked
6 CINSSEN	Card Insertion Status Enable: 1 Enabled 0 Masked
5 BRRSEN	Buffer Read Ready Status Enable: 1 Enabled 0 Masked
4 BWRSEN	Buffer Write Ready Status Enable: 1 Enabled 0 Masked
3 DINTSEN	DMA Interrupt Status Enable: 1 Enabled 0 Masked
2 BGESEN	Block Gap Event Status Enable: 1 Enabled 0 Masked
1 TCSEN	Transfer Complete Status Enable: 1 Enabled 0 Masked
0 CCSEN	Command Complete Status Enable: 1 Enabled 0 Masked

67.8.15 Interrupt Signal Enable (uSDHCx_INT_SIGNAL_EN)

This register is used to select which interrupt status is indicated to the Host System as the interrupt. These status bits all share the same interrupt line. Setting any of these bits to 1 enables interrupt generation. The corresponding Status register bit will generate an interrupt when the corresponding interrupt signal enable bit is set.

Address: Base address + 38h offset



uSDHCx_INT_SIGNAL_EN field descriptions

Field	Description
31–29 Reserved	This read-only field is reserved and always has the value 0.
28 DMAEIEN	DMA Error Interrupt Enable: 1 Enable 0 Masked
27 Reserved	This read-only field is reserved and always has the value 0.
26 TNEIEN	Tuning Error Interrupt Enable 1 Enabled 0 Masked
25 Reserved	This read-only field is reserved and always has the value 0.
24 AC12EIEN	Auto CMD12 Error Interrupt Enable: 1 Enabled 0 Masked
23 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

uSDHCx_INT_SIGNAL_EN field descriptions (continued)

Field	Description
22 DEBEIEN	Data End Bit Error Interrupt Enable: 1 Enabled 0 Masked
21 DCEIEN	Data CRC Error Interrupt Enable: 1 Enabled 0 Masked
20 DTOEIEN	Data Timeout Error Interrupt Enable: 1 Enabled 0 Masked
19 CIEIEN	Command Index Error Interrupt Enable: 1 Enabled 0 Masked
18 CEBEIEN	Command End Bit Error Interrupt Enable: 1 Enabled 0 Masked
17 CCEIEN	Command CRC Error Interrupt Enable: 1 Enabled 0 Masked
16 CTOEIEN	Command Timeout Error Interrupt Enable 1 Enabled 0 Masked
15 Reserved	This read-only field is reserved and always has the value 0.
14 TPIEN	Tuning Pass Interrupt Enable 1 Enabled 0 Masked
13 Reserved	This read-only field is reserved and always has the value 0.
12 RTEIEN	Re-Tuning Event Interrupt Enable 1 Enabled 0 Masked
11–9 Reserved	This read-only field is reserved and always has the value 0.
8 CINTIEN	Card Interrupt Interrupt Enable: 1 Enabled 0 Masked
7 CRMIEN	Card Removal Interrupt Enable:

Table continues on the next page...

uSDHCx_INT_SIGNAL_EN field descriptions (continued)

Field	Description
	1 Enabled 0 Masked
6 CINSIEN	Card Insertion Interrupt Enable: 1 Enabled 0 Masked
5 BRRIEN	Buffer Read Ready Interrupt Enable: 1 Enabled 0 Masked
4 BWRIEN	Buffer Write Ready Interrupt Enable: 1 Enabled 0 Masked
3 DINTIEN	DMA Interrupt Enable: 1 Enabled 0 Masked
2 BGEIEN	Block Gap Event Interrupt Enable: 1 Enabled 0 Masked
1 TCIEN	Transfer Complete Interrupt Enable: 1 Enabled 0 Masked
0 CCIEN	Command Complete Interrupt Enable: 1 Enabled 0 Masked

67.8.16 Auto CMD12 Error Status (uSDHCx_AUTOCMD12_ERR_STATUS)

When the Auto CMD12 Error Status bit in the Status register is set, the Host Driver shall check this register to identify what kind of error the Auto CMD12 / CMD 23 indicated. Auto CMD23 errors are indicated in bit 04-01.This register is valid only when the Auto CMD12 Error status bit is set.

The table bwlow shows the relationship between the Auto CMGD12 CRC Error and the Auto CMD12 Command Timeout Error.

Table 67-69. Relationship Between Command CRC Error and Command Timeout Error for Auto CMD12

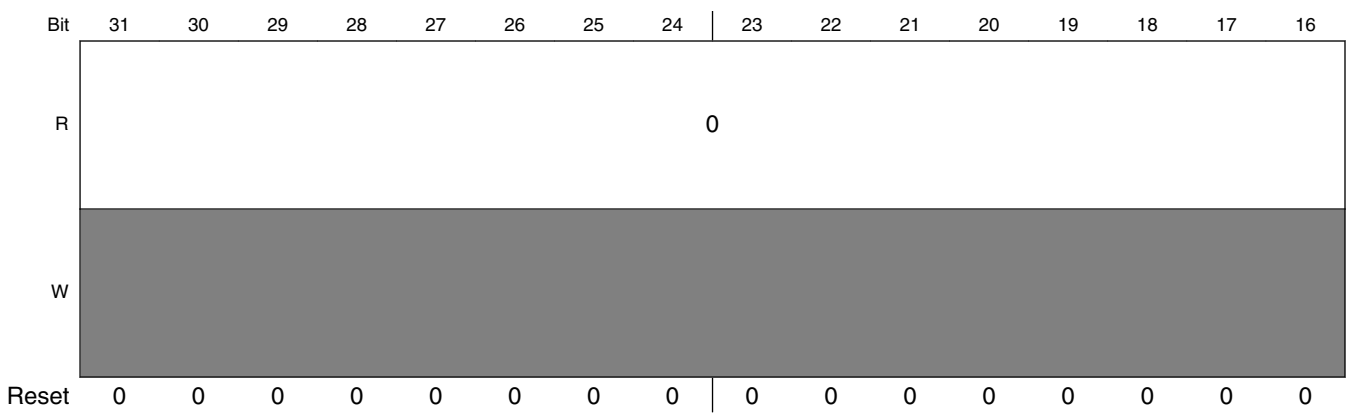
Auto CMD12 CRC Error	Auto CMD12 Timeout Error	Type of Error
0	0	No Error
0	1	Response Timeout Error
1	0	Response CRC Error
1	1	CMD line conflict

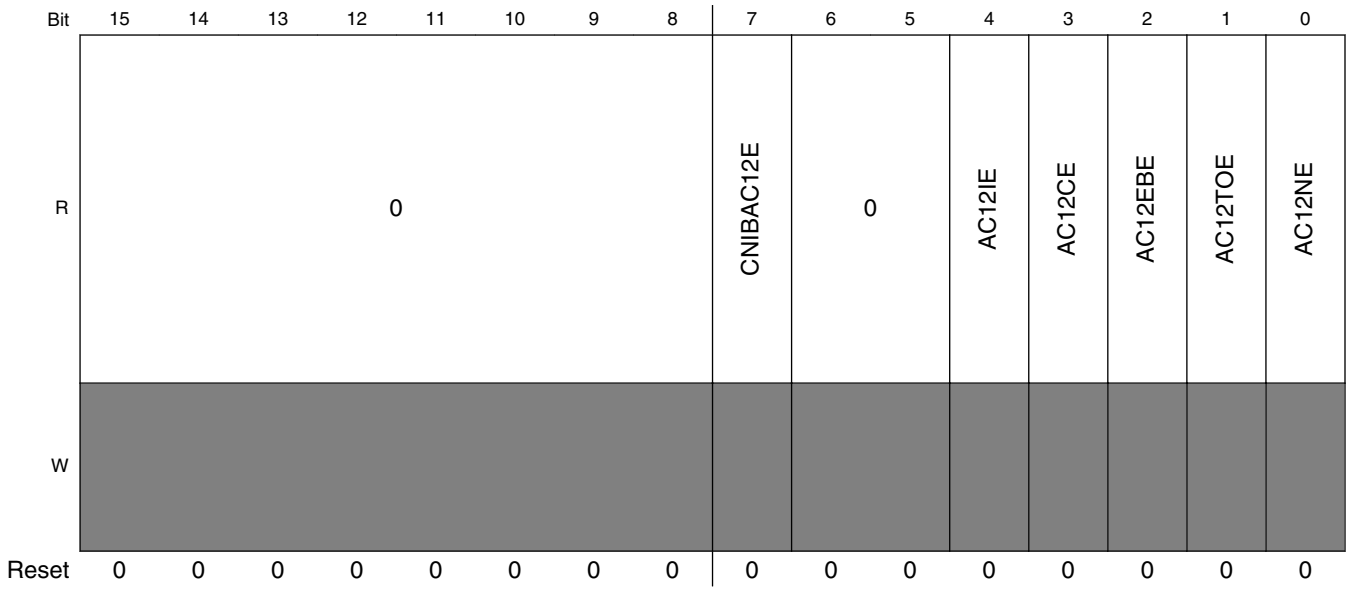
Changes in Auto CMD12 Error Status register can be classified in three scenarios:

1. When the uSDHC is going to issue an Auto CMD12.
   - Set bit 0 to 1 if the Auto CMD12 can't be issued due to an error in the previous command
   - Set bit 0 to 0 if the Auto CMD12 is issued
2. At the end bit of an Auto CMD12 response.
   - Check errors correspond to bits 1-4.
   - Set bits 1-4 corresponding to detected errors.
   - Clear bits 1-4 corresponding to detected errors
3. Before reading the Auto CMD12 Error Status bit 7.
   - Set bit 7 to 1 if there is a command that can't be issued
   - Clear bit 7 if there is no command to issue

The timing for generating the Auto CMD12 Error and writing to the Command register are asynchronous. After that, bit 7 shall be sampled when the driver is not writing to the Command register. So it is suggested to read this register only when the AC12E bit in Interrupt Status register is set. An Auto CMD12 Error Interrupt is generated when one of the error bits (0-4) is set to 1. The Command Not Issued By Auto CMD12 Error does not generate an interrupt.

Address: Base address + 3Ch offset





uSDHCx_AUTOCMD12_ERR_STATUS field descriptions

Field	Description
31–8 Reserved	This read-only field is reserved and always has the value 0.
7 CNIBAC12E	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. 1 Not Issued 0 No error
6–5 Reserved	This read-only field is reserved and always has the value 0.
4 AC12IE	Auto CMD12/23 Index Error: Occurs if the Command Index error occurs in response to a command. 1 Error, the CMD index in response is not CMD12/23 0 No error
3 AC12CE	Auto CMD12/23 CRC Error: Occurs when detecting a CRC error in the command response. 1 CRC Error Met in Auto CMD12/23 Response 0 No CRC error
2 AC12EBE	Auto CMD12/23 End Bit Error: Occurs when detecting that the end bit of command response is 0 which should be 1. 1 End Bit Error Generated 0 No error
1 AC12TOE	Auto CMD12/23 Timeout Error: Occurs if no response is returned within 64 SDCLK cycles from the end bit of the command. If this bit is set to1, the other error status bits (2-4) have no meaning.

Table continues on the next page...

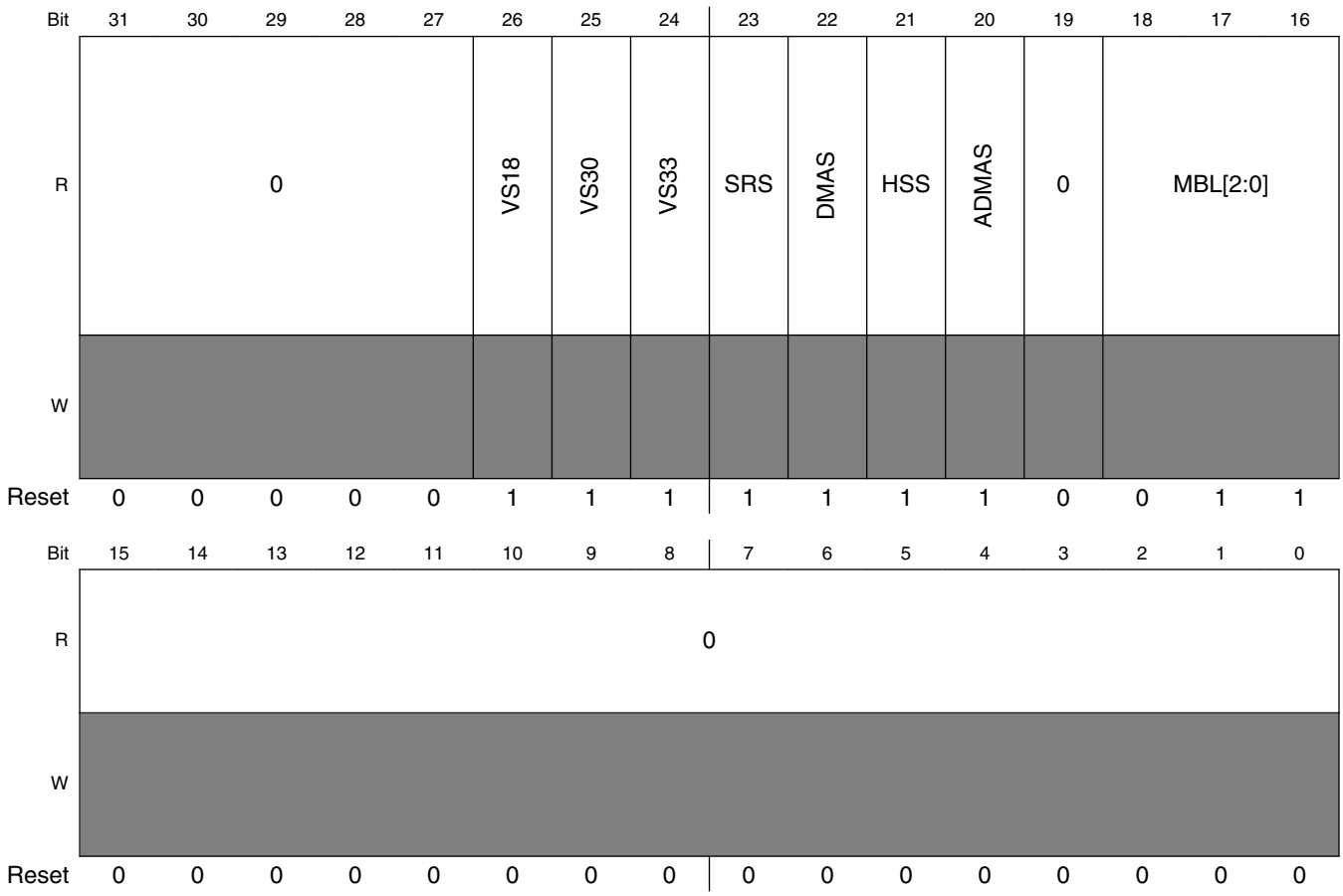
uSDHCx_AUTOCMD12_ERR_STATUS field descriptions (continued)

Field	Description
	1 Time out 0 No error
0 AC12NE	Auto CMD12 Not Executed: If memory multiple block data transfer is not started, due to a command error, this bit is not set because it is not necessary to issue an Auto CMD12. Setting this bit to 1 means the uSDHC cannot issue the Auto CMD12 to stop a memory multiple block data transfer due to some error. If this bit is set to 1, other error status bits (1-4) have no meaning. 1 Not executed 0 Executed

67.8.17 Host Controller Capabilities (uSDHCx_HOST_CTRL_CAP)

This register provides the Host Driver with information specific to the uSDHC implementation. The value in this register is the power-on-reset value, and does not change with a software reset.

Address: Base address + 40h offset



uSDHCx_HOST_CTRL_CAP field descriptions

Field	Description
31–27 Reserved	This read-only field is reserved and always has the value 0.
26 VS18	Voltage Support 1.8V: This bit shall depend on the Host System ability. 1 1.8V supported 0 1.8V not supported

Table continues on the next page...

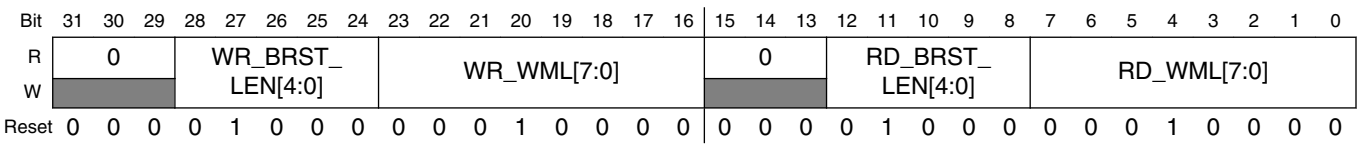
uSDHCx_HOST_CTRL_CAP field descriptions (continued)

Field	Description
25 VS30	Voltage Support 3.0V: This bit shall depend on the Host System ability. 1 3.0V supported 0 3.0V not supported
24 VS33	Voltage Support 3.3V: This bit shall depend on the Host System ability. 1 3.3V supported 0 3.3V not supported
23 SRS	Suspend / Resume Support: This bit indicates whether the uSDHC supports Suspend / Resume functionality. If this bit is 0, the Suspend and Resume mechanism, as well as the Read Wait, are not supported, and the Host Driver shall not issue either Suspend or Resume commands. 1 Supported 0 Not supported
22 DMAS	DMA Support: This bit indicates whether the uSDHC is capable of using the internal DMA to transfer data between system memory and the data buffer directly. 1 DMA Supported 0 DMA not supported
21 HSS	High Speed Support: This bit indicates whether the uSDHC supports High Speed mode and the Host System can supply a SD Clock frequency from 25 MHz to 50 MHz. 1 High Speed Supported 0 High Speed Not Supported
20 ADMAS	ADMA Support: This bit indicates whether the uSDHC supports the ADMA feature. 1 Advanced DMA Supported 0 Advanced DMA Not supported
19 Reserved	This read-only field is reserved and always has the value 0.
18–16 MBL[2:0]	Max Block Length: This value indicates the maximum block size that the Host Driver can read and write to the buffer in the uSDHC. The buffer shall transfer block size without wait cycles. 000 512 bytes 001 1024 bytes 010 2048 bytes 011 4096 bytes
Reserved	This read-only field is reserved and always has the value 0.

67.8.18 Watermark Level (uSDHCx_WTMK_LVL)

Both write and read watermark levels (FIFO threshold) are configurable. There value can range from 1 to 128 words. Both write and read burst lengths are also Configurable. There value can range from 1 to 31 words.

Address: Base address + 44h offset



uSDHCx_WTMK_LVL field descriptions

Field	Description
31–29 Reserved	This read-only field is reserved and always has the value 0.
28–24 WR_BRST_ LEN[4:0]	Write Burst Length: [1] The number of words the uSDHC writes in a single burst. The write burst length must be less than or equal to the write watermark level, and all bursts within a watermark level transfer will be in back-to-back mode. On reset, this field will be 8. Writing 0 to this field will result in '01000' (i.e. it is not able to clear this field).
23–16 WR_WML[7:0]	Write Watermark Level: The number of words used as the watermark level (FIFO threshold) in a DMA write operation. Also the number of words as a sequence of write bursts in back-to-back mode. The maximum legal value for the write watermark level is 128.
15–13 Reserved	This read-only field is reserved and always has the value 0.
12–8 RD_BRST_ LEN[4:0]	Read Burst Length: [2] The number of words the uSDHC reads in a single burst. The read burst length must be less than or equal to the read watermark level, and all bursts within a watermark level transfer will be in back-to-back mode. On reset, this field will be 8. Writing 0 to this field will result in '01000' (i.e. it is not able to clear this field).
RD_WML[7:0]	Read Watermark Level: The number of words used as the watermark level (FIFO threshold) in a DMA read operation. Also the number of words as a sequence of read bursts in back-to-back mode. The maximum legal value for the read water mark level is 128.

1. Due to system restriction, the actual burst length may not exceed 16.
2. Due to system restriction, the actual burst length may not exceed 16.

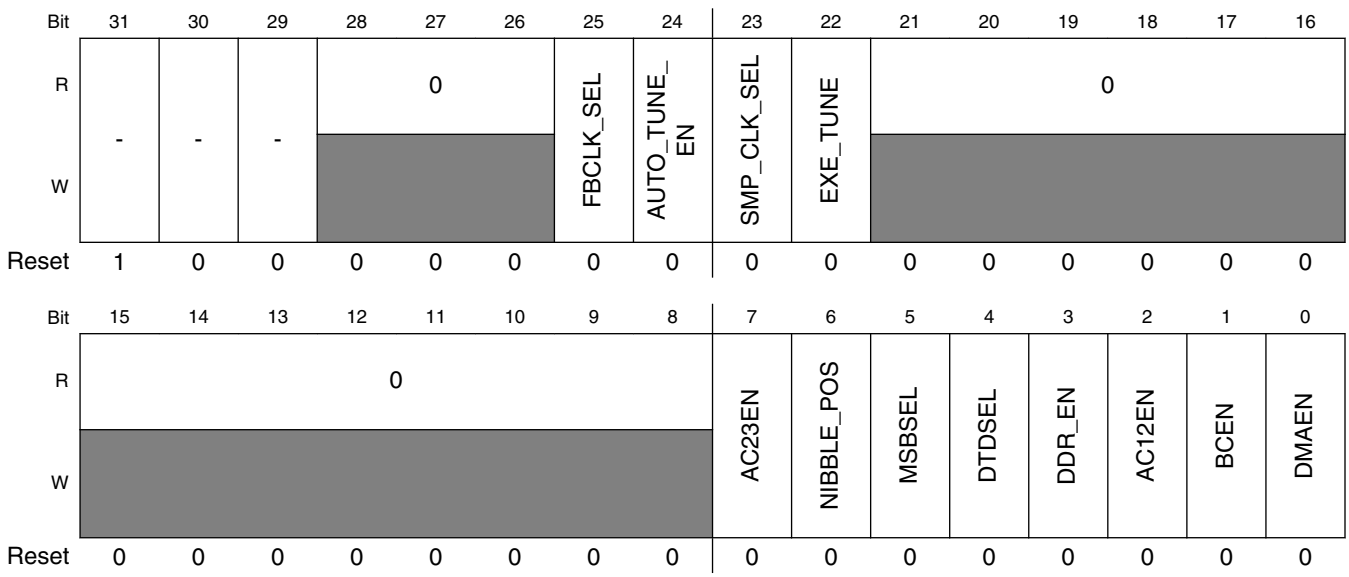
67.8.19 Mixer Control (uSDHCx_MIX_CTRL)

This register is used to DMA and data transfer. To prevent data loss, The software should check if data transfer is active before writing to this register. These bits are DPSEL, MBSEL, DTDSEL, AC12EN, BCEN and DMAEN.

Table 67-73. Transfer Type Register Setting for Various Transfer Types

Multi/Single Block Select	Block Count Enable	Block Count	Function
0	Don't Care	Don't Care	Single Transfer
1	0	Don't Care	Infinite Transfer
1	1	Positive Number	Multiple Transfer
1	1	Zero	No Data Transfer

Address: Base address + 48h offset



uSDHCx_MIX_CTRL field descriptions

Field	Description
31 -	Reserved. Always write as 1
30 -	Reserved. Always write as 0.
29 -	Reserved. Always write as 0.
28–26 Reserved	This read-only field is reserved and always has the value 0.
25 FBCLK_SEL	Feedback clock source selection (Only used for SD3.0, SDR104 mode)

Table continues on the next page...

uSDHCx_MIX_CTRL field descriptions (continued)

Field	Description
	1 feedback clock comes from the ipp_card_clk_out 0 feedback clock comes from the loopback CLK
24 AUTO_TUNE_EN	Auto tuning enable (Only used for SD3.0, SDR104 mode) 1 enable auto tuning 0 disable auto tuning
23 SMP_CLK_SEL	Tuned clock or Fixed clock is used to sample data/cmd (Only used for SD3.0, SDR104 mode) 1 Tuned clock is used to sample data/cmd 0 Fixed clock is used to sample data/cmd
22 EXE_TUNE	Execute Tuning: (Only used for SD3.0, SDR104 mode) This bit is set to 1 to indicate the Host Driver is starting tuning procedure. Tuning procedure is aborted by writing 0. 1 Execute Tuning 0 Not Tuned or Tuning Completed
21–8 Reserved	This read-only field is reserved and always has the value 0.
7 AC23EN	Auto CMD23 Enable When this bit is set to 1, the Host Controller issues a CMD23 automatically before issuing a command specified in the Command Register.
6 NIBBLE_POS	In DDR 4-bit mode nibble position indictation. 0- the sequence is 'odd high nibble -> even high nibble -> odd low nibble -> even low nibble'; 1- the sequence is 'odd high nibble -> odd low nibble -> even high nibble -> even low nibble'.
5 MSBSEL	Multi / Single Block Select: This bit enables multiple block DATA line data transfers. 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. (Refer to Command Transfer Type (uSDHC_CMD_XFR_TYP)). 1 Multiple Blocks 0 Single Block
4 DTDSEL	Data Transfer Direction Select: This bit defines the direction of DATA line data transfers. The bit is set to 1 by the Host Driver to transfer data from the SD card to the uSDHC and is set to 0 for all other commands. 1 Read (Card to Host) 0 Write (Host to Card)
3 DDR_EN	Dual Data Rate mode selection
2 AC12EN	Auto CMD12 Enable: Multiple block transfers for memory require a CMD12 to stop the transaction. When this bit is set to 1, the uSDHC will issue a CMD12 automatically when the last block transfer has completed. The Host Driver shall not set this bit to issue commands that do not require CMD12 to stop a multiple block data transfer. In particular, secure commands defined in File Security Specification (see reference list) do not require CMD12. In single block transfer, the uSDHC will ignore this bit no matter if it is set or not. 1 Enable 0 Disable

Table continues on the next page...

uSDHCx_MIX_CTRL field descriptions (continued)

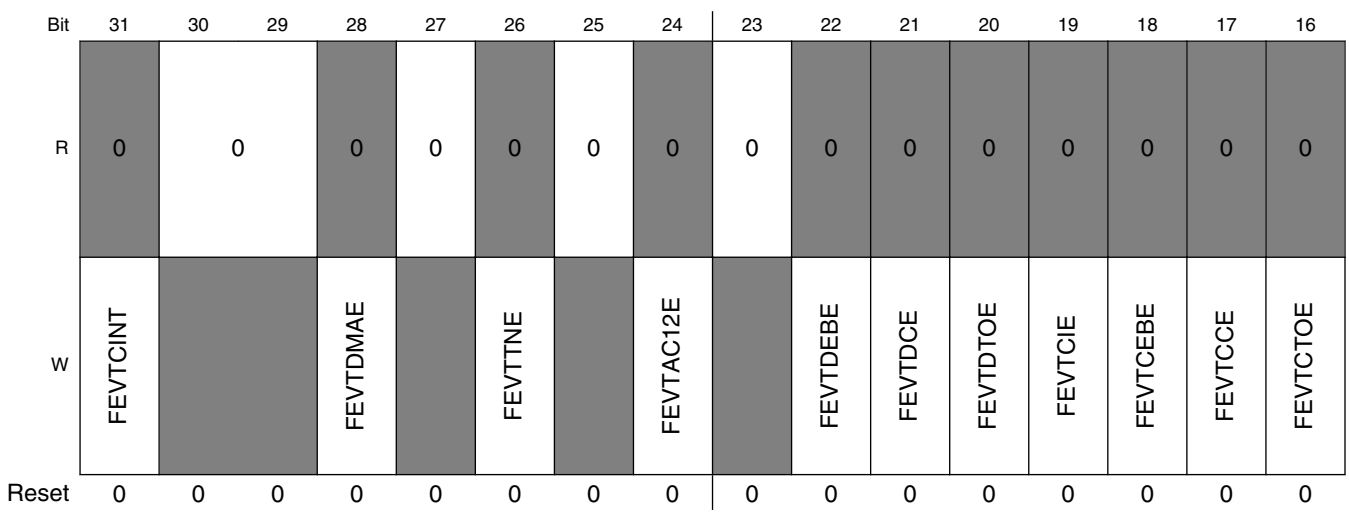
Field	Description
1 BCEN	Block Count Enable: This bit is used to enable the Block Count register, which is only relevant for multiple block transfers. When this bit is 0, the internal counter for block is disabled, which is useful in executing an infinite transfer. 1 Enable 0 Disable
0 DMAEN	DMA Enable: This bit enables DMA functionality. If this bit is set to 1, a DMA operation shall begin when the Host Driver sets the DPSEL bit of this register. Whether the Simple DMA, or the Advanced DMA, is active depends on the DMA Select field of the Protocol Control register. 1 Enable 0 Disable

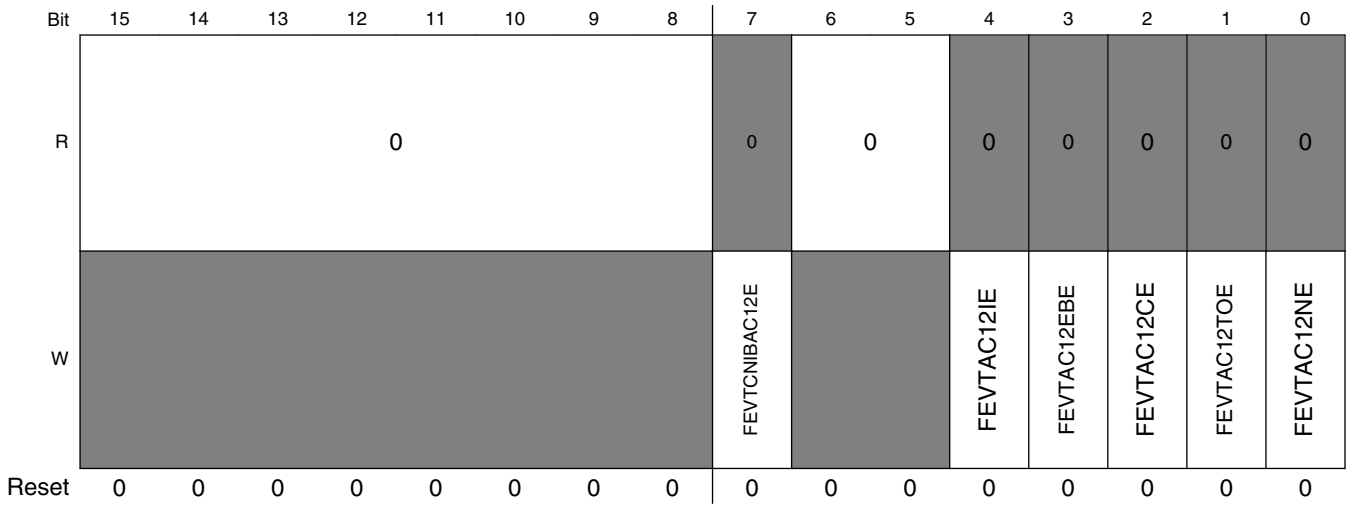
67.8.20 Force Event (uSDHCx_FORCE_EVENT)

The Force Event Register is not a physically implemented register. Rather, it is an address at which the Interrupt Status Register can be written if the corresponding bit of the Interrupt Status Enable Register is set. This register is a write only register and writing 0 to it has no effect. Writing 1 to this register actually sets the corresponding bit of Interrupt Status Register. A read from this register always results in 0's. In order to change the corresponding status bits in the Interrupt Status Register, make sure to set IPGEN bit in System Control Register so that ipg_clk is always active.

Forcing a card interrupt will generate a short pulse on the DATA1 line, and the driver may treat this interrupt as a normal interrupt. The interrupt service routine may skip polling the card interrupt factor as the interrupt is self cleared.

Address: Base address + 50h offset





uSDHC*x*_FORCE_EVENT field descriptions

Field	Description
31 FEVTCINT	Force Event Card Interrupt: Writing 1 to this bit generates a short low-level pulse on the internal DATA1 line, as if a self clearing interrupt was received from the external card. If enabled, the CINT bit will be set and the interrupt service routine may treat this interrupt as a normal interrupt from the external card.
30–29 Reserved	This read-only field is reserved and always has the value 0.
28 FEVTDMAE	Force Event DMA Error: Forces the DMAE bit of Interrupt Status Register to be set
27 Reserved	This read-only field is reserved and always has the value 0.
26 FEVTTNE	Force Tuning Error: Forces the TNE bit of Interrupt Status Register to be set
25 Reserved	This read-only field is reserved and always has the value 0.
24 FEVTAC12E	Force Event Auto Command 12 Error: Forces the AC12E bit of Interrupt Status Register to be set
23 Reserved	This read-only field is reserved and always has the value 0.
22 FEVTDEBE	Force Event Data End Bit Error: Forces the DEBE bit of Interrupt Status Register to be set
21 FEVTDCE	Force Event Data CRC Error: Forces the DCE bit of Interrupt Status Register to be set
20 FEVTDTOE	Force Event Data Time Out Error: Force the DTOE bit of Interrupt Status Register to be set
19 FEVTCIE	Force Event Command Index Error: Forces the CCE bit of Interrupt Status Register to be set
18 FEVTCEBE	Force Event Command End Bit Error:

Table continues on the next page...

uSDHCx_FORCE_EVENT field descriptions (continued)

Field	Description
	Forces the CEBE bit of Interrupt Status Register to be set
17 FEVTCCE	Force Event Command CRC Error: Forces the CCE bit of Interrupt Status Register to be set
16 FEVTCTOE	Force Event Command Time Out Error: Forces the CTOE bit of Interrupt Status Register to be set
15–8 Reserved	This read-only field is reserved and always has the value 0.
7 FEVTCNIBAC12E	Force Event Command Not Executed By Auto Command 12 Error: Forces the CNIBAC12E bit in the Auto Command12 Error Status Register to be set
6–5 Reserved	This read-only field is reserved and always has the value 0.
4 FEVTAC12IE	Force Event Auto Command 12 Index Error: Forces the AC12IE bit in the Auto Command12 Error Status Register to be set
3 FEVTAC12EBE	Force Event Auto Command 12 End Bit Error: Forces the AC12EBE bit in the Auto Command12 Error Status Register to be set
2 FEVTAC12CE	Force Event Auto Command 12 CRC Error: Forces the AC12CE bit in the Auto Command12 Error Status Register to be set
1 FEVTAC12TOE	Force Event Auto Command 12 Time Out Error: Forces the AC12TOE bit in the Auto Command12 Error Status Register to be set
0 FEVTAC12NE	Force Event Auto Command 12 Not Executed: Forces the AC12NE bit in the Auto Command12 Error Status Register to be set

67.8.21 ADMA Error Status Register (uSDHCx_ADMA_ERR_STATUS)

When an ADMA Error Interrupt has occurred, the ADMA Error States field in this register holds the ADMA state and the ADMA System Address register holds the address around the error descriptor.

For recovering from this error, the Host Driver requires the ADMA state to identify the error descriptor address as follows:

- ST_STOP: Previous location set in the ADMA System Address register is the error descriptor address
- ST_FDS: Current location set in the ADMA System Address register is the error descriptor address

- ST_CADR: This state is never set because it only increments the descriptor pointer and doesn't generate an ADMA error
- ST_TFR: Previous location set in the ADMA System Address register is the error descriptor address

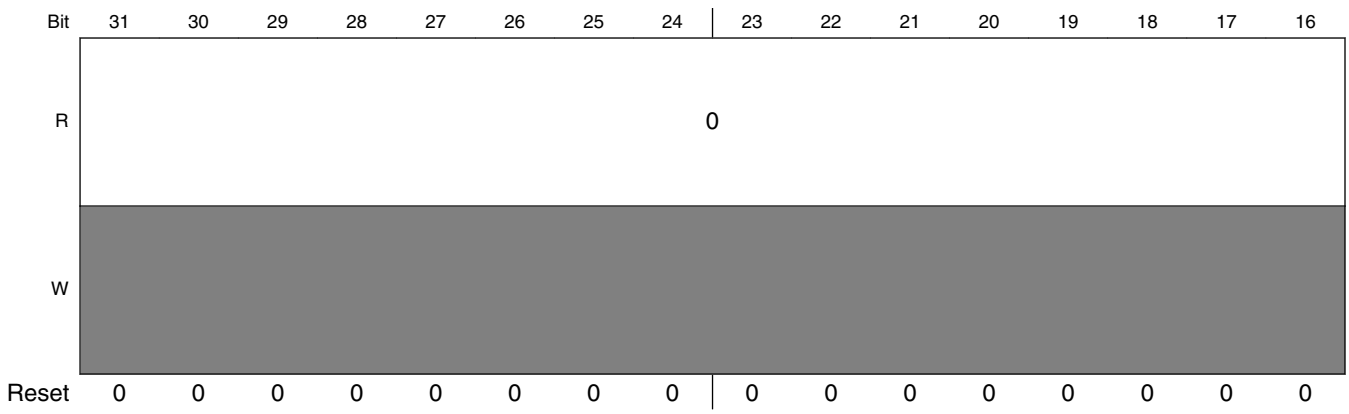
In case of a write operation, the Host Driver should use the ACMD22 to get the number of the written block, rather than using this information, since unwritten data may exist in the Host Controller.

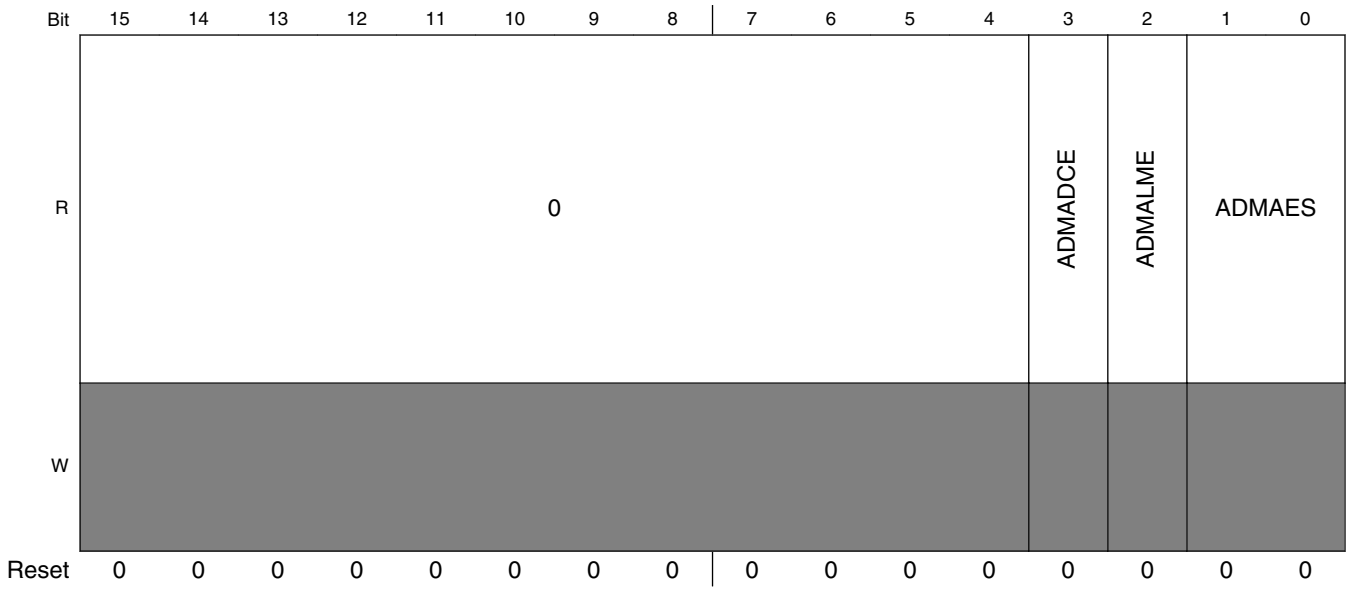
The Host Controller generates the ADMA Error Interrupt when it detects invalid descriptor data (Valid=0) in the ST_FDS state. The Host Driver can distinguish this error by reading the Valid bit of the error descriptor.

Table 67-76. ADMA Error State Coding

D01-D00	ADMA Error State (when error has occurred)	Contents of ADMA System Address Register
00	ST_STOP (Stop DMA)	Holds the address of the next executable Descriptor command
01	ST_FDS (Fetch Descriptor)	Holds the valid Descriptor address
10	ST_CADR (Change Address)	No ADMA Error is generated
11	ST_TFR (Transfer Data)	Holds the address of the next executable Descriptor command

Address: Base address + 54h offset





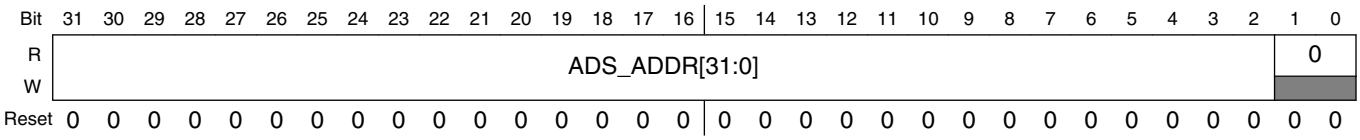
uSDHCx_ADMA_ERR_STATUS field descriptions

Field	Description
31–4 Reserved	This read-only field is reserved and always has the value 0.
3 ADMADCE	ADMA Descritor Error: This error occurs when invalid descriptor fetched by ADMA: 1 Error 0 No Error
2 ADMALME	ADMA Length Mismatch Error: This error occurs in the following 2 cases: • While the Block Count Enable is 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 can not be divided by the block length 1 Error 0 No Error
ADMAES	ADMA Error State (when ADMA Error is occurred.): This field indicates the state of the ADMA when an error has occurred during an ADMA data transfer. Refer to ADMA Error Status Register (uSDHC_ADMA_ERR_STATUS) for more details.

67.8.22 ADMA System Address (uSDHCx_ADMA_SYS_ADDR)

This register contains the physical system memory address used for ADMA transfers.

Address: Base address + 58h offset



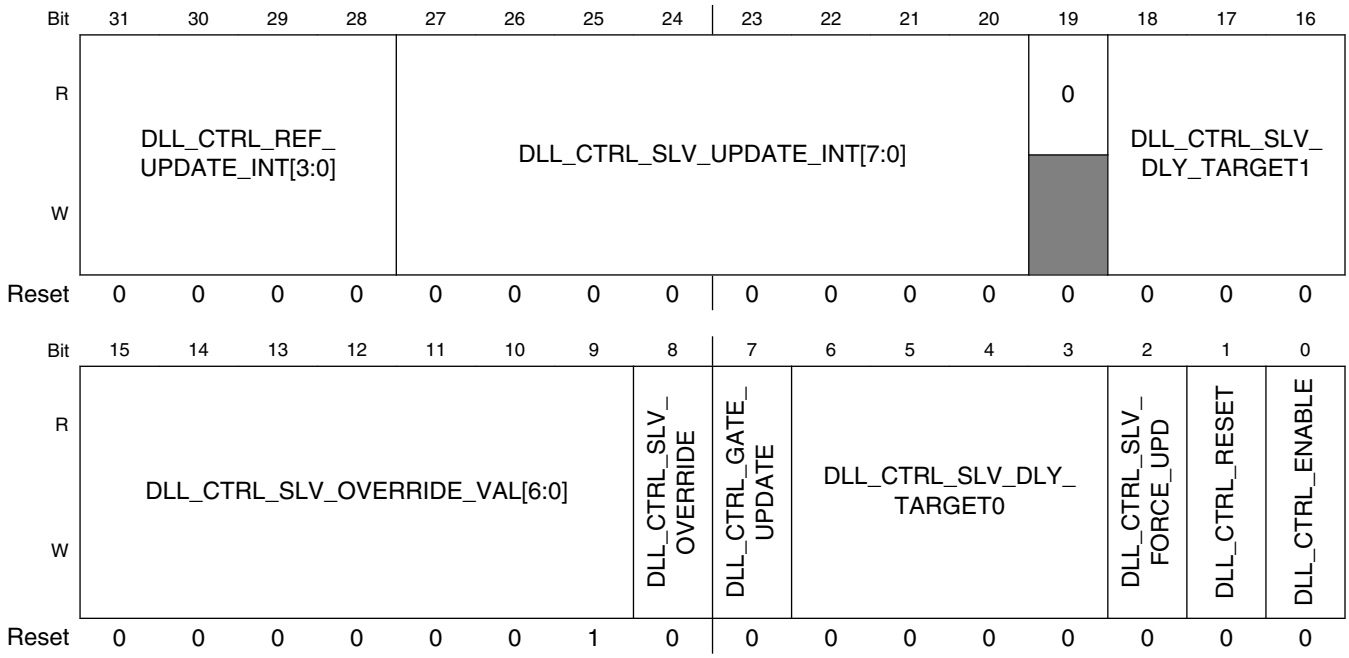
uSDHCx_ADMA_SYS_ADDR field descriptions

Field	Description
31–2 ADS_ ADDR[31:0]	ADMA System Address: This register holds the word address of the executing command in the Descriptor table. At the start of ADMA, the Host Driver shall set the start address of the Descriptor table. The ADMA engine increments this register address whenever fetching a Descriptor command. When the ADMA is stopped at the Block Gap, this register indicates the address of the next executable Descriptor command. When the ADMA Error Interrupt is generated, this register shall hold the valid Descriptor address depending on the ADMA state. The lower 2 bits of this register is tied to '0' so the ADMA address is always word aligned. Since this register supports dynamic address reflecting, when TC bit is set, it automatically alters the value of internal address counter, so SW cannot change this register when TC bit is set. Such restriction is also listed in Software Restrictions .
Reserved	This read-only field is reserved and always has the value 0.

67.8.23 DLL (Delay Line) Control (uSDHCx_DLL_CTRL)

This register contains control bits for DLL.

Address: Base address + 60h offset



uSDHCx_DLL_CTRL field descriptions

Field	Description
31–28 DLL_CTRL_REF_UPDATE_INT[3:0]	DLL control loop update interval. The interval cycle is (2 + REF_UPDATE_INT) * ref_clock. By default, the DLL control loop shall update every two ref_clock cycles. It should be noted that increasing the reference delay-line update interval reduces the ability of the DLL to adjust to fast changes in conditions that may effect the delay (such as voltage and temperature)
27–20 DLL_CTRL_SLV_UPDATE_INT[7:0]	Slave delay line update interval. If default 0 is used, it means 256 cycles of ref_clock. A value of 0x0f results in 15 cycles and so on. Note that software can always cause an update of the slave-delay line using the SLV_FORCE_UPDATE register. Note that the slave delay line will also update automatically when the reference DLL transitions to a locked state (from an un-locked state).
19 Reserved	This read-only field is reserved and always has the value 0.
18–16 DLL_CTRL_SLV_DLY_TARGET1	Refer to DLL_CTRL_SLV_DLY_TARGET0 below.
15–9 DLL_CTRL_SLV_OVERRIDE_VAL[6:0]	When SLV_OVERRIDE=1 This field is used to select 1 of 128 physical taps manually. A value of 0 selects tap 1, and a value of 0x7f selects tap 128.

Table continues on the next page...

uSDHCx_DLL_CTRL field descriptions (continued)

Field	Description
8 DLL_CTRL_ SLV_OVERRIDE	Set this bit to 1 to Enable manual override for slave delay chain using SLV_OVERRIDE_VAL; to set 0 to disable manual override. This feature does not require the DLL to be enabled using the ENABLE bit. In fact to reduce power, if SLV_OVERRIDE is used, it is recommended to disable the DLL with ENABLE=0
7 DLL_CTRL_ GATE_UPDATE	Set this bit to 1 to prevent the DLL from updating (since when clock_in exists, glitches may appear during DLL updates). This bit may be used by software if such a condition occurs. Clear the bit to 0 to allow the DLL to update automatically.
6–3 DLL_CTRL_ SLV_DLY_ TARGET0	The delay target for the uSDHC loopback read clock can be programmed in 1/16th increments of an ref_clock half-period. The delay is ((DLL_CTRL_SLV_DLY_TARGET1 +1)* ref_clock/2)/16 So the input read-clock can be delayed relative input data from (ref_clock/2)/16 to ref_clock*4
2 DLL_CTRL_ SLV_FORCE_ UPD	Setting this bit to 1, forces the slave delay line to update to the DLL calibrated value immediately. The slave delay line shall update automatically based on the SLV_UPDATE_INT interval or when a DLL lock condition is sensed. Subsequent forcing of the slave-line update can only occur if SLV_FORCE_UP is set back to 0 and then asserted again (edge triggered). Be sure to use it when uSDHC is idle. This function may not work when uSDHC is working on data/cmd/response.
1 DLL_CTRL_ RESET	Setting this bit to 1 force a reset on DLL. This will cause the DLL to lose lock and re-calibrate to detect an ref_clock half period phase shift. This signal is used by the DLL as edge-sensitive, so in order to create a subsequent reset, RESET must be taken low and then asserted again
0 DLL_CTRL_ ENABLE	Set this bit to 1 to enable the DLL and delay chain; otherwise; set to 0 to bypasses DLL. Note that using the slave delay line override feature with SLV_OVERRIDE and SLV_OVERRIDE VAL, the DLL does not need to be enabled

67.8.24 DLL Status (uSDHC*x*_DLL_STATUS)

This register contains the DLL status information. All bits are read only and will read the same as the power-reset value.

Address: Base address + 64h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_STS_REF_SEL[6:0]								DLL_STS_SLV_SEL[6:0]						DLL_STS_REF_LOCK	DLL_STS_SLV_LOCK
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

uSDHCx_DLL_STATUS field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15–9 DLL_STS_REF_SEL[6:0]	Reference delay line select taps. This is encoded by 7 bits for 127 taps.
8–2 DLL_STS_SLV_SEL[6:0]	Slave delay line select status. This is the instant value generated from reference chain. Since the reference chain can only be updated when ref_clock is detected, this value should be the right value to be updated when the reference is locked.
1 DLL_STS_REF_LOCK	Reference DLL lock status. This signifies that the DLL has detected and locked to a half-phase ref_clock shift, allowing the slave delay-line to perform programmed clock delays
0 DLL_STS_SLV_LOCK	Slave delay-line lock status. This signifies that a valid calibration has been set to the slave-delay line and that the slave-delay line is implementing the programmed delay value

67.8.25 CLK Tuning Control and Status (uSDHCx_CLK_TUNE_CTRL_STATUS)

This register contains the Clock Tuning Control status information. All bits are read only and will read the same as the power-reset value. This register is added to support SD3.0 UHS-I SDR104 mode.

Address: Base address + 68h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PRE_ERR	TAP_SEL_PRE[6:0]							TAP_SEL_OUT[3:0]				TAP_SEL_POST[3:0]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NXT_ERR	DLY_CELL_SET_PRE[6:0]							DLY_CELL_SET_OUT[6:0]				DLY_CELL_SET_POST[6:0]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

uSDHCx_CLK_TUNE_CTRL_STATUS field descriptions

Field	Description
31 PRE_ERR	PRE error which means the number of delay cells added on the feedback clock is too small. It's valid only when SMP_CLK_SEL of Mix control register(bit23 of 0x48) is enabled.

Table continues on the next page...

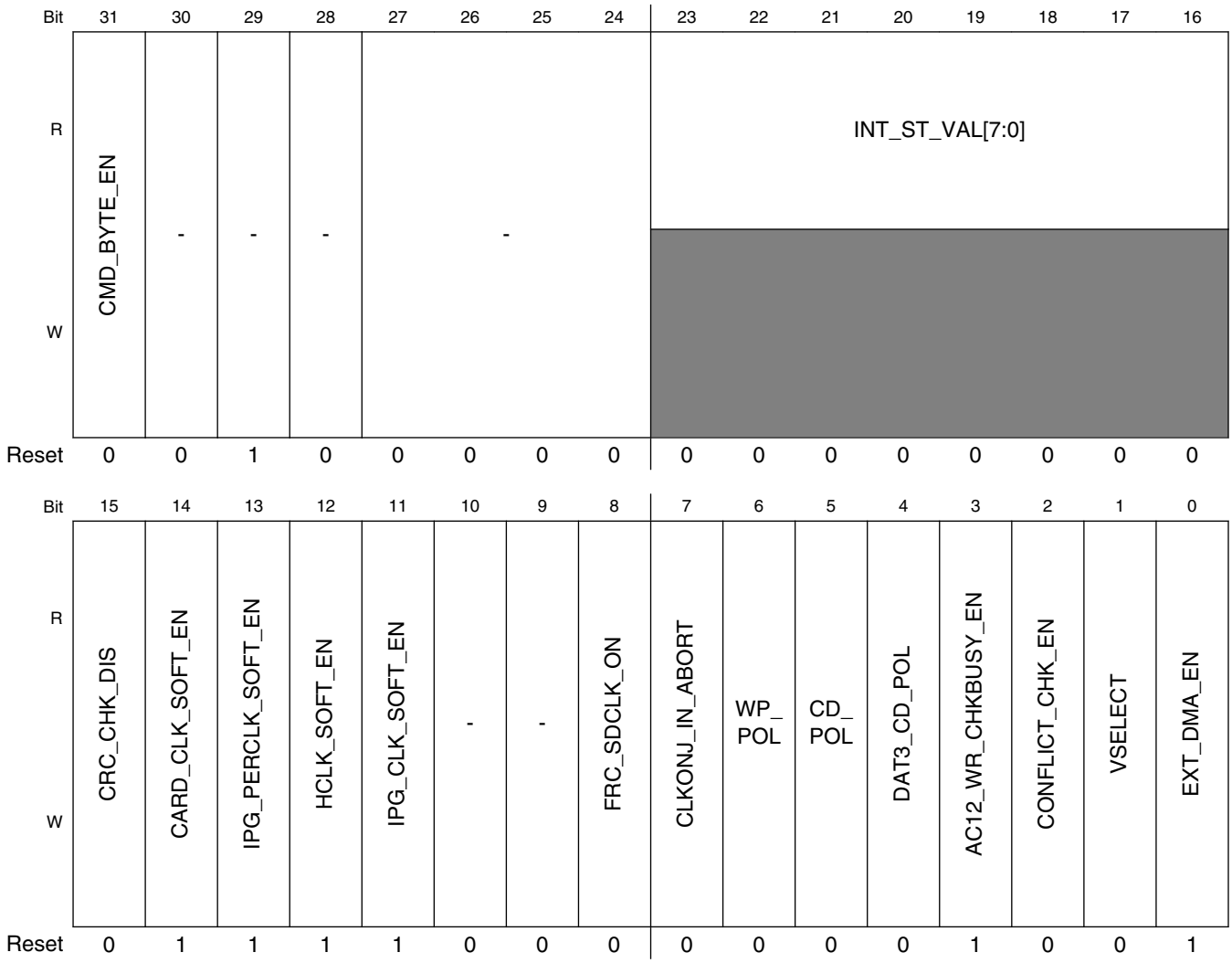
uSDHCx_CLK_TUNE_CTRL_STATUS field descriptions (continued)

Field	Description
30–24 TAP_SEL_ PRE[6:0]	Reflects the number of delay cells added on the feedback clock between the feedback clock and CLK_PRE. When AUTO_TUNE_EN(bit24 of 0x48) is disabled, TAP_SEL_PRE is always equal to DLY_CELL_SET_PRE. When AUTO_TUNE_EN(bit24 of 0x48) is enabled, TAP_SEL_PRE will be updated automatically according to the status of the auto tuning circuit to adjust the sample clock phase.
23–20 TAP_SEL_ OUT[3:0]	Reflect the number of delay cells added on the feedback clock between CLK_PRE and CLK_OUT.
19–16 TAP_SEL_ POST[3:0]	Reflect the number of delay cells added on the feedback clock between CLK_OUT and CLK_POST.
15 NXT_ERR	NXT error which means the number of delay cells added on the feedback clock is too large. It's valid only when SMP_CLK_SEL of Mix control register(bit23 of 0x48) is enabled.
14–8 DLY_CELL_ SET_PRE[6:0]	Set the number of delay cells on the feedback clock between the feedback clock and CLK_PRE.
7–4 DLY_CELL_ SET_OUT[6:0]	Set the number of delay cells on the feedback clock between CLK_PRE and CLK_OUT.
DLY_CELL_ SET_POST[6:0]	Set the number of delay cells on the feedback clock between CLK_OUT and CLK_POST.

67.8.26 Vendor Specific Register (uSDHCx_VEND_SPEC)

This register contains the vendor specific control/status register.

Address: Base address + C0h offset



uSDHCx_VEND_SPEC field descriptions

Field	Description
31 CMD_BYTE_EN	byte access 0 Disable 1 Enable
30 -	Reserved. Always write as 0.

Table continues on the next page...

uSDHCx_VEND_SPEC field descriptions (continued)

Field	Description
29 -	Reserved. Always write as 1.
28 -	Reserved. Always write as 0.
27–24 -	Reserved. Always write as 4'b0000.
23–16 INT_ST_VAL[7:0]	Internal State Value Internal state value, reflecting the corresponding state value selected by Debug Select field. This field is read-only and write to this field does not have effect.
15 CRC_CHK_DIS	CRC check disable 0 check CRC16 for every read data packet and check CRC bits for every write data packet 1 ignore CRC16 check for every read data packet and ignore CRC bits check for every write data packet
14 CARD_CLK_ SOFT_EN	card clock software enable 0 gate off the sd_clk 1 enable the sd_clk
13 IPG_PERCLK_ SOFT_EN	ipg_perclk software enable 0 gate off the ipg_perclk 1 enable the ipg_perclk
12 HCLK_SOFT_EN	Please note, hardware auto-enables the AHB clock when the internal DMA is enabled even if HCLK_SOFT_EN is 0. AHB clock software enable 0 gate off the AHB clock. 1 enable the AHB clock.
11 IPG_CLK_ SOFT_EN	IPG_CLK software enable 0 gate off the IPG_CLK 1 enable the IPG_CLK
10 -	Reserved. Always write as 0.
9 -	Reserved. Always write as 0.
8 FRC_SDCLK_ ON	Force CLK output active: 0 CLK active or inactive is fully controlled by the hardware 1 force CLK active
7 CLKONJ_IN_ ABORT	Only for debug. Force CLK output active when sending Abort command: 0 the CLK output is active when sending abort command while data is transmitting even if the internal FIFO is full(for read) or empty(for write) 1 the CLK output is inactive when sending abort command while data is transmitting if the internal FIFO is full(for read) or empty(for write)
6 WP_POL	Only for debug.

Table continues on the next page...

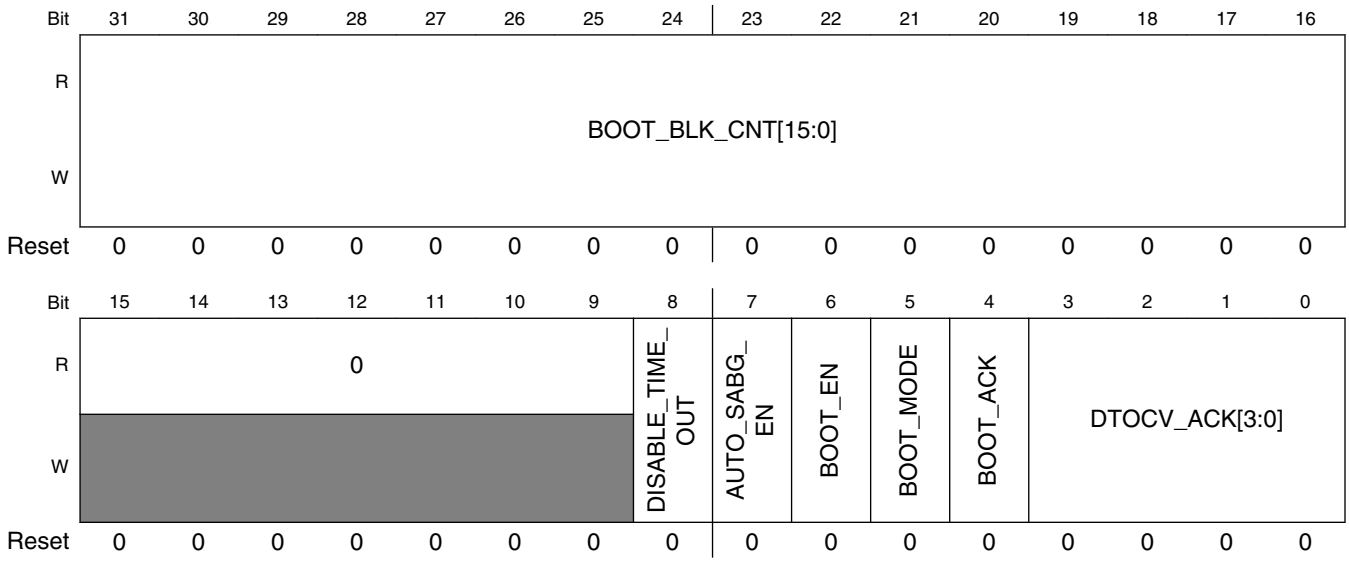
uSDHC*x*_VEND_SPEC field descriptions (continued)

Field	Description
	Polarity of the WP pin: 0 WP pin is high active 1 WP pin is low active
5 CD_POL	Only for debug. Polarity of the CD_B pin: 0 CD_B pin is low active 1 CD_B pin is high active
4 DAT3_CD_POL	Only for debug. Polarity of DATA3 pin when it's used as card detection: 0 card detected when DATA3 is high 1 card detected when DATA3 is low
3 AC12_WR_ CHKBUSY_EN	Check busy enable after auto CMD12 for write data packet 0 Do not check busy after auto CMD12 for write data packet 1 Check busy after auto CMD12 for write data packet
2 CONFLICT_ CHK_EN	It's not implemented in uSDHC IP. Conflict check enable. 0 conflict check disable 1 conflict check enable
1 VSELECT	Voltage Selection Change the value of output signal VSELECT, to control the voltage on pads for external card. There must be a control circuit out of uSDHC to change the voltage on pads. 1 Change the voltage to low voltage range, around 1.8V 0 Change the voltage to high voltage range, around 3.0V
0 EXT_DMA_EN	External DMA Request Enable Enable the request to external DMA. When the internal DMA (either Simple DMA or Advanced DMA) is not in use, and this bit is set, uSDHC will send out DMA request when the internal buffer is ready. This bit is particularly useful when transferring data by ARM platform polling mode, and it is not allowed to send out the external DMA request. By default, this bit is set. 0 In any scenario, uSDHC does not send out external DMA request 1 When internal DMA is not active, the external DMA request will be sent out

67.8.27 MMC Boot Register (uSDHC*x*_MMC_BOOT)

This register contains the MMC Fast Boot control register.

Address: Base address + C4h offset



uSDHC*x*_MMC_BOOT field descriptions

Field	Description
31–16 BOOT_BLK_CNT[15:0]	The value defines the Stop At Block Gap value of automatic mode. When received card block cnt is equal to (BLK_CNT - BOOT_BLK_CNT) and AUTO_SABG_EN is 1, then Stop At Block Gap. Here, BLK_CNT is defined in the Block Atrributes Register, bit31-16 of 0x04.
15–9 Reserved	This read-only field is reserved and always has the value 0.
8 DISABLE_TIME_OUT	Please note, when this bit is set, there is no timeout check no matter whether boot_en is set or not. Disable time out. 0 Enable time out 1 Disable time out
7 AUTO_SABG_EN	During boot, enable auto stop at block gap function. This function will be triggered, and host will stop at block gap when received card block cnt is equal to (BLK_CNT - BOOT_BLK_CNT).
6 BOOT_EN	Boot mode enable. 0 Fast boot disable 1 Fast boot enable
5 BOOT_MODE	Boot mode select. 0 Normal boot 1 Alternative boot

Table continues on the next page...

uSDHCx_MMC_BOOT field descriptions (continued)

Field	Description
4 BOOT_ACK	Boot ack mode select. 0 No ack 1 Ack
DTOCV_ ACK[3:0]	Boot ACK time out counter value. 0000 SDCLK x 2^13 0001 SDCLK x 2^14 0010 SDCLK x 2^15 0011 SDCLK x 2^16 0100 SDCLK x 2^17 0101 SDCLK x 2^18 0110 SDCLK x 2^19 0111 SDCLK x 2^20 1110 SDCLK x 2^27 1111 SDCLK x 2^28

67.8.28 Vendor Specific 2 Register (uSDHCx_VEND_SPEC2)

This register contains the vendor specific control 2 register.

Address: Base address + C8h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0						0	0	CARD_INT_AUTO_CLR_DIS	TUNING_CMD_EN	TUNING_1bit_EN	TUNING_8bit_EN	CARD_INT_D3_TEST	SDR104_NSD_DIS	SDR104_OE_DIS	SDR104_TIMING_DIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

uSDHCx_VEND_SPEC2 field descriptions

Field	Description
31–10 Reserved	This read-only field is reserved and always has the value 0.
9 Reserved	This read-only field is reserved and always has the value 0.
8 Reserved	This read-only field is reserved and always has the value 0.
7 CARD_INT_ AUTO_CLR_DIS	Disable the feature to clear the Card interrupt status bit when Card Interrupt status enable bit is cleared. Only for debug. 0 Card interrupt status bit(CINT) can be cleared when Card Interrupt status enable bit is 0. 1 Card interrupt status bit(CINT) can only be cleared by writting a 1 to CINT bit.
6 TUNING_CMD_ EN	Enable the auto tuning circuit to check the CMD line. 0 Auto tuning circuit doesn't check the CMD line. 1 Auto tuning circuit checks the CMD line.
5 TUNING_1bit_EN	Enable the auto tuning circuit to check the DATA0 only. It's used with the TUNING_8bit_EN together.
4 TUNING_8bit_EN	Enable the auto tuning circuit to check the DATA[7:0]. It's used with the TUNING_1bit_EN together. **NOTE:** The format of these two bits are [TUNNING_8bit_EN:TUNNING_1bit_EN]. 00 Tuning circuit only checks the DATA[3:0]. 01 Tuning circuit only checks the DATA0. 10 Tuning circuit checks the whole DATA[7:0]. 11 Invalid.
3 CARD_INT_D3_ TEST	Card interrupt detection test. This bit only uses for debugging. 0 Check the card interrupt only when DATA3 is high. 1 Check the card interrupt by ignoring the status of DATA3.
2 SDR104_NSD_ DIS	Interrupt window after abort command is sent. This bit only uses for debugging. 0 Enable the interrupt window 9 cycles later after the end of the I/O abort command(or CMD12) is sent. 1 Enable the interrupt window 5 cycles later after the end of the I/O abort command(or CMD12) is sent.
1 SDR104_OE_ DIS	CMD_OE/DATA_OE logic generation test. This bit only uses for debugging. 0 Drive the CMD_OE/DATA_OE for one more clock cycle after the end bit. 1 Stop to drive the CMD_OE/DATA_OE at once after driving the end bit.
0 SDR104_ TIMING_DIS	Timeout counter test. This bit only uses for debugging. 0 The timeout counter for Ncr changes to 80, Ncrc changes to 21. 1 The timeout counter for Ncr changes to 72, Ncrc changes to 15.

Chapter 68
Video Data Order Adapter (VDOA)

68.1 Overview

The function of the VDOA is to reorder video data from the "tiled" order used by the VPU to the conventional raster-scan order needed by the IPU .

68.1.1 Block Diagram

The following figure shows the VDOA top level block diagram.

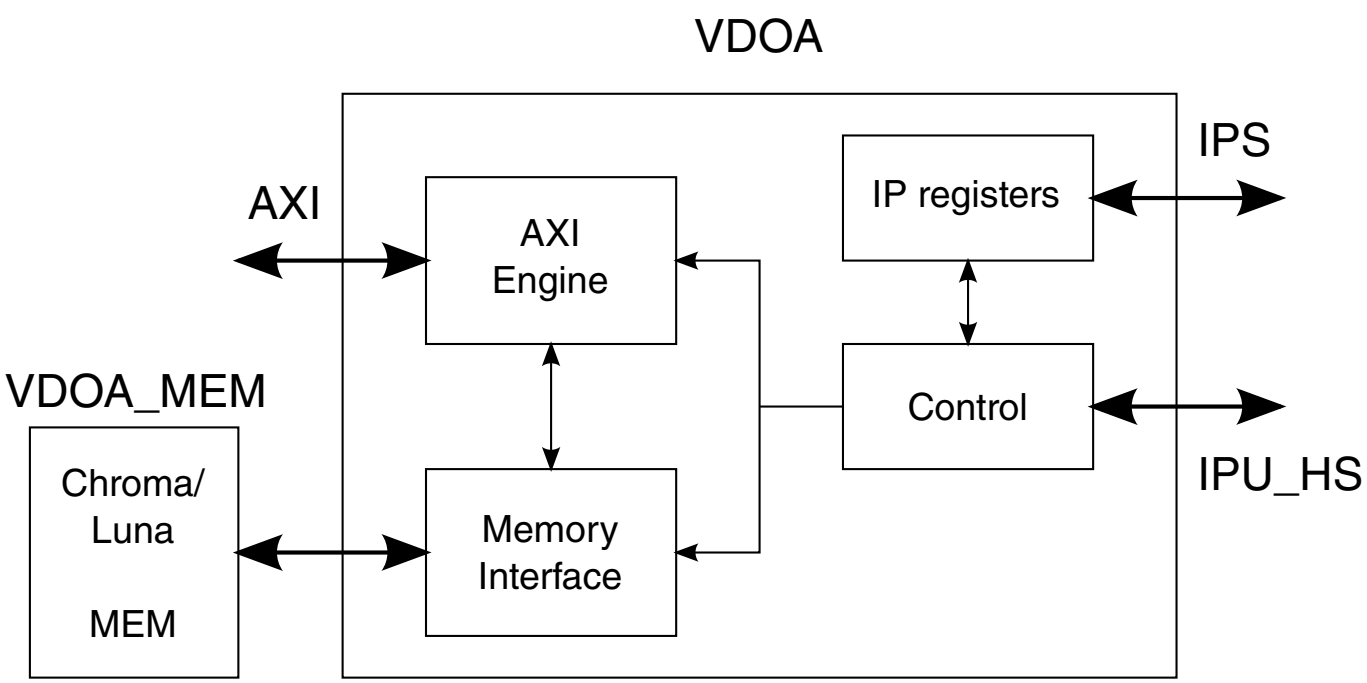


Figure 68-1. VDOA Block Diagram

68.1.2 Features

- Works with 266-MHz clock
- Converts tiled macro-blocks to raster-scan order
- Output formats of IPU data 4:2:0 partially interleaved and 4:2:2 interleaved, format of VPU input data 4:2:0 partially interleaved
- Total rate of 3 or 2.4 pixel/clk, depending on IPU data format (see Data Rate)
- Frame width: even; up to 8192 pixels
- Frame height: even; up to 4096 pixels
- Addresses (bytes)
  - Base address: 32-bit integer, multiple of 8
  - Offset of Chroma buffer from the Lumma buffer : 27-bit integer, multiple of 8
  - IPU stride line: 14-bit integer, multiple of 8
  - VPU stride line: 13-bit integer, multiple of 8
- Each VDOA task is activated individually by the CPU and consists of reordering a single buffer of pixels; or reordering concurrently three buffers of pixels (needed for the three input fields used for de-interlacing in the IPU).
- All task parameters are double buffered to allow software-controlled frame-by-frame task switching without reconfiguration overhead.
- Ability to decode full frame continuously or a band (a few lines) at a time, synchronized with the IPU using double buffer for output data
- Ability to work with one of two IPU units
- Interrupt on task completion and on AXI transfer error

68.2 Clocks

The table found here describes the clock sources for VDOA.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 68-1. VDOA Clocks

Clock name	Clock Root	Description
ipg_clk_s	vdo_axi_clk_root	IP bus clock domain. Controls VDOA registers read/write function.
vdoa_clk	vdo_axi_clk_root	VDOA functional clock. Maximum frequency of 266MHz. The actual value of the core clock is dependent upon the maximum use case. Clock frequency scaling will be done through the VPU API.

68.3 Functional Description

This section provides a complete functional description of the block.

68.3.1 Memory organization

68.3.1.1 Frame organization - VPU

The data of the VPU will always be in YUV 4:2:0 partially interleaved format.

Data is organized in macro blocks which correspond to 16x16 pixel blocks in the frame. Each macro block is divided into two 8x16 blocks.

A cluster of four pixels in 4:2:0 format is composed of four luma values (Y, 1 byte) and one chroma value (Cr, Cb, 2 bytes). In partially interleaved format, the luma and chroma exist on sperate buffers. The chroma buffer is 16x8 bytes in size.

The following table shows an example of how data is arranged in a frame and in blocks:

Table 68-2. FW*FH frame Luma buffer Example

0/0-7	0/8-15	0/16-23	0/24-31	...	0/16x-16x+7	0/16x+8-16x+15	...	0/FW-16-FW-9	0/FW-8-FW-1
1/0-7	1/8-15	1/16-23	1/24-31	...	1/16x-16x+7	1/16x+8-16x+15	...	1/FW-16-FW-9	1/FW-8-FW-1
...	...	...	...	...	...	...	...	...	...
15/0-7	15/8-15	15/16-23	15/24-32	...	15/16x-16x+7	15/16x+8-16x+15	...	15/FW-16-FW-9	15/FW-8-FW-1
16/0-7	16/8-15	16/16-23	16/24-31	...	16/16x-16x+7	16/16x+8-16x+15	...	16/FW-16-FW-9	16/FW-8-FW-1
17/0-7	17/8-15	17/16-23	17/24-31	...	17/16x-16x+7	17/16x+8-16x+15	...	17/FW-16-FW-9	17/FW-8-FW-1
...	...	...	...	...	...	...	...	...	...
31/0-7	31/8-15	31/16-23	31/24-32	...	31/16x-16x+7	31/16x+8-16x+15	...	31/FW-16-FW-9	31/FW-8-FW-1
...	...	...	...	...	...	...	...	...	...
FH-16/0-7	FH-16/8-15	FH-16/16-23	FH-16/24-31	...	FH-16/16x-16x+7	FH-16/16x+8-16x+15	...	FH-16/FW-16-FW-9	FH-16/FW-8-FW-1
FH-15/0-7	FH-15/8-15	FH-15/16-23	FH-15/24-31	...	FH-15/16x-16x+7	FH-15/16x+8-16x+15	...	FH-15/FW-16-FW-9	FH-15/FW-8-FW-1
...	...	...	...	...	...	...	...	...	...
FH-1/0-7	FH-1/8-15	FH-1/16-23	FH-1/24-31	...	FH-1/16x-16x+7	FH-1/16x+8-16x+15	...	FH-1/FW-16-FW-9	FH-1/FW-8-FW-1

Legend -> row/Start pixel - End pixel

68.3.1.2 Frame organization - IPU

The IPU support various frame organization (or pixel format). VDOA supports only two of these formats: 4:2:0 partially interleaved (similar to VPU) and 4:2:2 interleaved.

68.3.2 Conversion types

The VDOA supports two types of data conversion:

1. VPU 4:2:0 partially interleaved -> IPU 4:2:0 partially interleaved
2. VPU 4:2:0 partially interleaved -> IPU 4:2:2 interleaved

NOTE

For all conversions the VPU format data is in tiled order and the IPU data is in raster order.

68.3.3 IPU synchronization

The synchronization with the IPU is described in %20%20%20%20%20%20%20%20%20/projects/MAD/block_guide_library/ipu/map_ipu.xml.

The synchronization between VDOA and IPU executed on band resolution (a band consists of several rows, depending on IPU's IDMAC configuration). Up to two band buffers are supported for this synchronization.

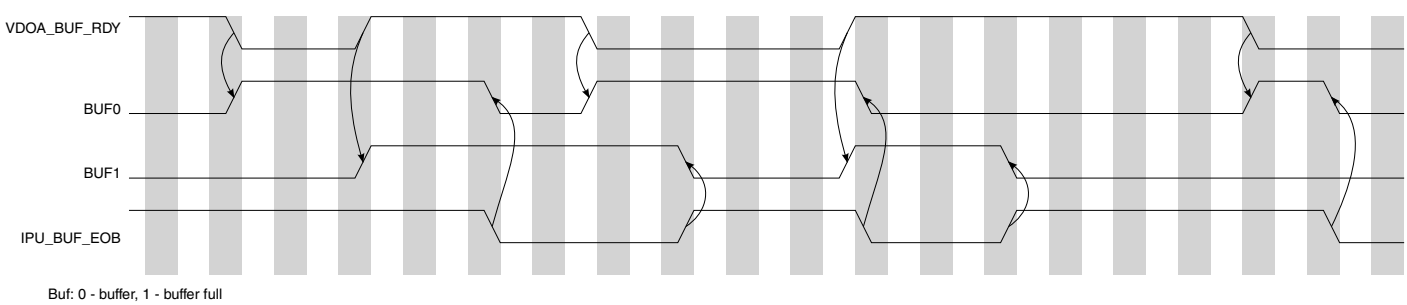


Figure 68-2. Toggling mechanism and buffer indication

68.3.4 Double buffering

All configuration registers (all registers excluding IVDOAE, VDOAIS, VDOAST, and VDOATD) are double buffered. This means that on start, their content is copied to internal registers.

Writing new settings to VDOA's configuration registers will affect the following transfer.

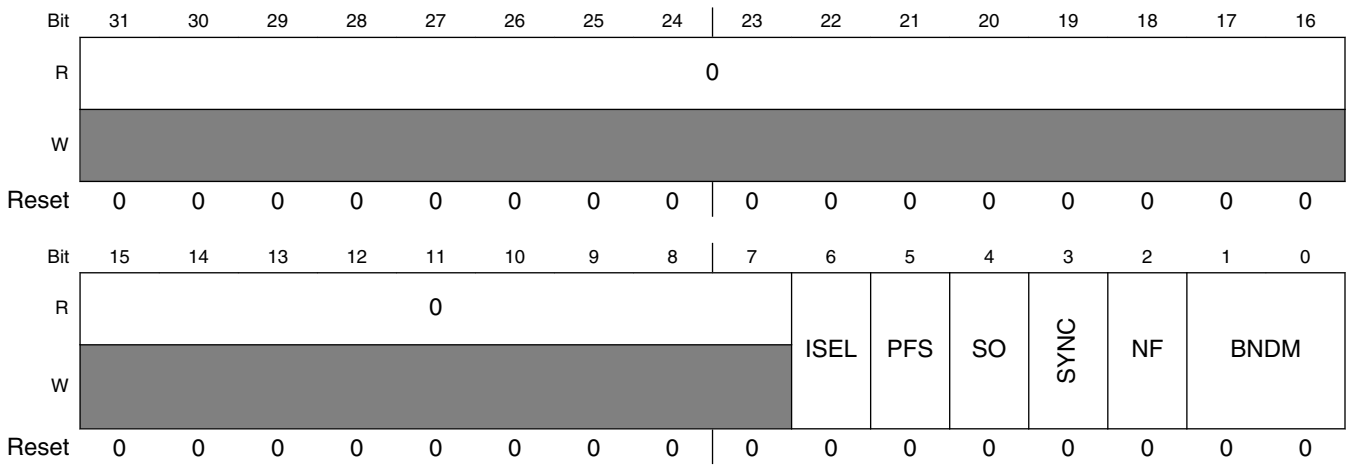
68.4 VDOA Memory Map/Register Definition

VDOA memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21E_4000	VDOA Control Register (VDOA_VDOAC)	32	R/W	0000_0000h	68.4.1/5694
21E_4004	VDOA Start and Reset (VDOA_VDOASRR)	32	R/W	0000_0000h	68.4.2/5695
21E_4008	VDOA Interrupt Enable Register (VDOA_VDOAIE)	32	R/W	0000_0000h	68.4.3/5696
21E_400C	VDOA Interrupt Status Register (VDOA_VDOAIST)	32	w1c	0000_0000h	68.4.4/5696
21E_4010	VDOA Frame Parameters Register (VDOA_VDOAFP)	32	R/W	0000_0000h	68.4.5/5697
21E_4014	VDOA IPU External Buffer 0 Frame 0 Address Register (VDOA_VDOAIEBA00)	32	R/W	0000_0000h	68.4.6/5697
21E_4018	VDOA IPU External Buffer 0 Frame 1 Address Register (VDOA_VDOAIEBA01)	32	R/W	0000_0000h	68.4.7/5698
21E_401C	VDOA IPU External Buffer 0 Frame 2 Address Register (VDOA_VDOAIEBA02)	32	R/W	0000_0000h	68.4.8/5698
21E_4020	VDOA IPU External Buffer 1 Frame 0 Address Register (VDOA_VDOAIEBA10)	32	R/W	0000_0000h	68.4.9/5698
21E_4024	VDOA IPU External Buffer 1 Frame 1 Address Register (VDOA_VDOAIEBA11)	32	R/W	0000_0000h	68.4.10/ 5699
21E_4028	VDOA IPU External Buffer 1 Frame 2 Address Register (VDOA_VDOAIEBA12)	32	R/W	0000_0000h	68.4.11/ 5699
21E_402C	VDOA IPU Stride Line Register (VDOA_VDOASL)	32	R/W	0000_0000h	68.4.12/ 5700
21E_4030	VDOA IPU U (Chroma) Buffer Offset Register (VDOA_VDOAIUBO)	32	R/W	0000_0000h	68.4.13/ 5700
21E_4034	VDOA VPU External Buffer 0 Address Register (VDOA_VDOAVEBA0)	32	R/W	0000_0000h	68.4.14/ 5701
21E_4038	VDOA VPU External Buffer 1 Address Register (VDOA_VDOAVEBA1)	32	R/W	0000_0000h	68.4.15/ 5701
21E_403C	VDOA VPU External Buffer 2 Address Register (VDOA_VDOAVEBA2)	32	R/W	0000_0000h	68.4.16/ 5701
21E_4040	VDOA VPU U (Chroma) Buffer Offset Register (VDOA_VDOAVUBO)	32	R/W	0000_0000h	68.4.17/ 5702
21E_4044	VDOA Status Register (VDOA_VDOASR)	32	R	0000_0000h	68.4.18/ 5703

68.4.1 VDOA Control Register (VDOA_VDOAC)

Address: 21E_4000h base + 0h offset = 21E_4000h



VDOA_VDOAC field descriptions

Field	Description
31–7 Reserved	This read-only field is reserved and always has the value 0.
6 ISEL	IPU SELECT - determines in sync mode which of the two sets of hand shake pins is used 0 **vdoa_buf_rdy_and_ipu_buf_eob_0** — Use vdoa_buf_rdy[0] and ipu_buf_eob[0] 1 **vdoa_buf_rdy_and_ipu_buf_eob_1** — Use vdoa_buf_rdy[1] and ipu_buf_eob[1]
5 PFS	Pixel Format Select - Pixel format of data written to / read from IPU. Note Data from VPU is always assumed to have partial interleaved 4:2:0 format 0 **4_2_0** — partialy interleaved 4:2:0 1 **4_2_2** — interleaved 4:2:2 Y1U1Y2V1
4 SO	Scan Order 0 **PROGRESSIVE** — Scan order is progressive 1 **INTERLACED** — Scan order is interlaced
3 SYNC	SYNC MODE - defines whether the VDOA will transfer a full frame (or 2 frames) continuously or will transfer a band at a time and wait for IPU signal to continue 0 **NO_SYNC_MODE** — None SYNC mode (default) 1 **SYNC_MODE** — Sync mode
2 NF	Number of frames - Determines whether to transfer 1 frame or three frames 0 **1_FRAME** — One frame (default) 1 **3_FRAMES** — Three frames
BNDM	BNDM Band Size 00 **BAND_HEIGHT_8** — Band height = 8 lines.- Supported only for interlaced scan (SO=1) 01 **BAND_HEIGHT_16** — Band height = 16 lines.

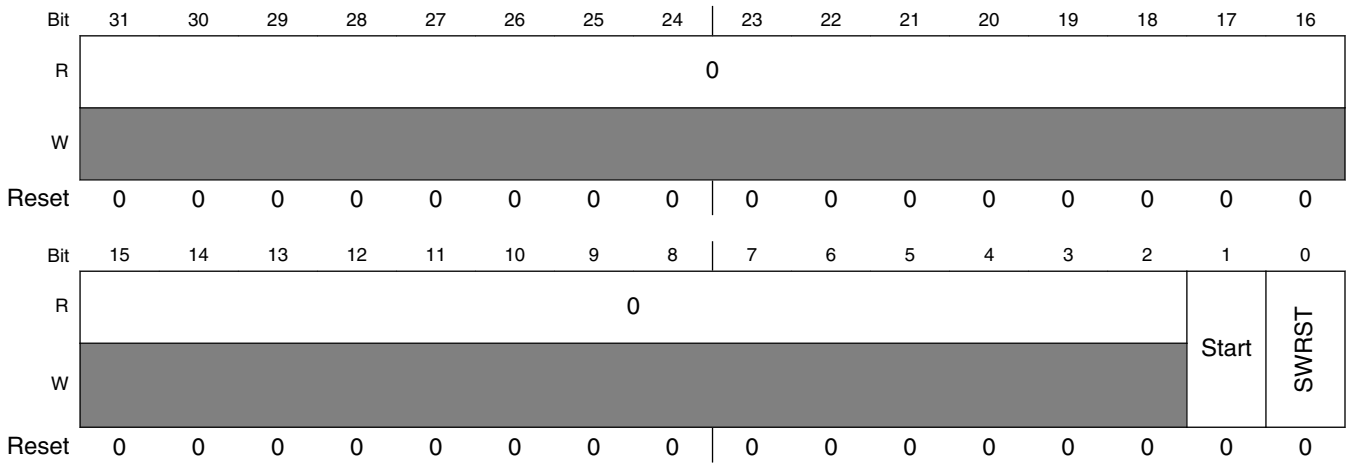
Table continues on the next page...

VDOA_VDOAC field descriptions (continued)

Field	Description
	10 **BAND_HEIGHT_32** — Band height = 32 lines. 11 reserved

68.4.2 VDOA Start and Reset (VDOA_VDOASRR)

Address: 21E_4000h base + 4h offset = 21E_4004h

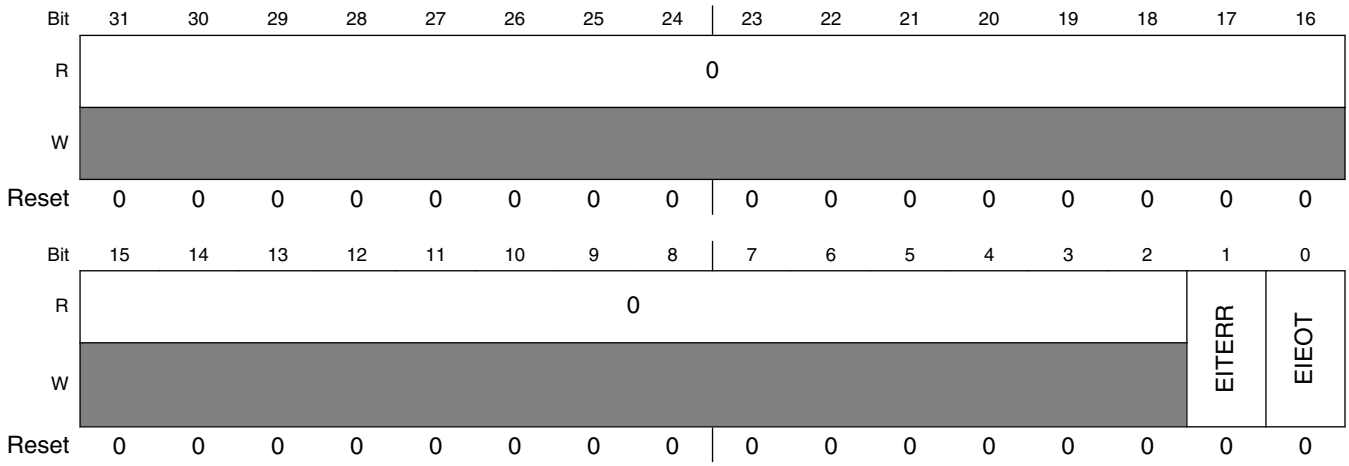


VDOA_VDOASRR field descriptions

Field	Description
31–2 Reserved	This read-only field is reserved and always has the value 0.
1 Start	Start Transfer - Strat a VDOA data transfer according to all parameters. **Note:** During run this bit is read only In IDLE - 0 Ignored 1 **START_TRANSFER** — Start a new transfer All registers we copied internally so any write to them will take place only in next transfer (double buffer)
0 SWRST	Software reset - Finish outstanding AXI transfer and reset all internal registers the configuration registers are mnot cleared

68.4.3 VDOA Interrupt Enable Register (VDOA_VDOAIE)

Address: 21E_4000h base + 8h offset = 21E_4008h

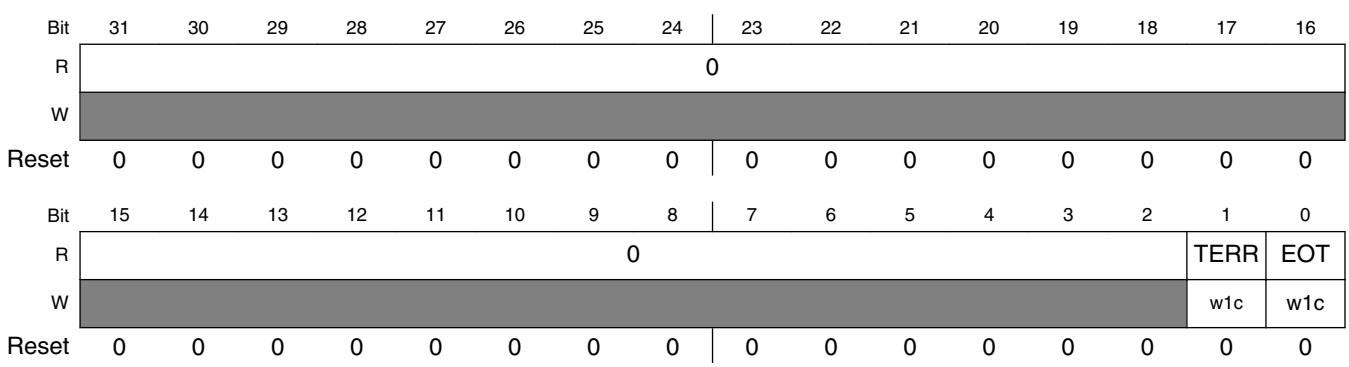


VDOA_VDOAIE field descriptions

Field	Description
31–2 Reserved	This read-only field is reserved and always has the value 0.
1 EITERR	EITERR - Enable Interrupt Transfer access Error - Enables Interrupt on AXI access Error 0 **IRQ_DISABLED** — interrupt disable (default) 1 **IRQ_ENABLED** — Interrupt Enabled
0 EIEOT	EIEOT - Enable Interrupt End Of Transfer- Enables Interrupt on end of transfer 0 **IRQ_DISABLED** — interrupt disable (default) 1 **IRQ_ENABLED** — Interrupt Enabled

68.4.4 VDOA Interrupt Status Register (VDOA_VDOAIST)

Address: 21E_4000h base + Ch offset = 21E_400Ch



VDOA_VDOAIST field descriptions

Field	Description
31–2 Reserved	This read-only field is reserved and always has the value 0.
1 TERR	Axi Access had an access error see ERRW bit in 0XBASE_0044 (VDOASR) for type of access (read write) if EITERR is set an interrupt will be generated
0 EOT	End Of transfer - Transfer was completed if EIEOT is set an interrupt will be generated

68.4.5 VDOA Frame Parameters Register (VDOA_VDOAFP)

Address: 21E_4000h base + 10h offset = 21E_4010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			FH													0		FW													
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

VDOA_VDOAFP field descriptions

Field	Description
31–29 Reserved	This read-only field is reserved and always has the value 0.
28–16 FH	Number of pixels in one column, of the frame. Note the 3 LSB are RO and will always be 0 (multiply of 8)
15–14 Reserved	This read-only field is reserved and always has the value 0.
FW	Number of pixels in one row, of the frame. Note the3 LSB are RO and will always be 0 (multiply of 8)

68.4.6 VDOA IPU External Buffer 0 Frame 0 Address Register (VDOA_VDOAIEBA00)

Address: 21E_4000h base + 14h offset = 21E_4014h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IEBA00																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

VDOA_VDOAIEBA00 field descriptions

Field	Description
IEBA00	External Address of Frame 0 output (IPU) buffer 0 - Note that the 3 LSB are always 0 (aligned to 8 address) Used for all transfer types

68.4.7 VDOA IPU External Buffer 0 Frame 1 Address Register (VDOA_VDOAIEBA01)

Address: 21E_4000h base + 18h offset = 21E_4018h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IEBA01																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

VDOA_VDOAIEBA01 field descriptions

Field	Description
IEBA01	External Address of Frame 1 output (IPU) buffer 0 - Note that the 3 LSB are always 0 (aligned to 8 address) Used when transfering 3 frames (NF=1) only

68.4.8 VDOA IPU External Buffer 0 Frame 2 Address Register (VDOA_VDOAIEBA02)

Address: 21E_4000h base + 1Ch offset = 21E_401Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IEBA02																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

VDOA_VDOAIEBA02 field descriptions

Field	Description
IEBA02	External Address of Frame 2 output (IPU) buffer 0 - Note that the 3 LSB are always 0 (aligned to 8 address) Used when transfering 3 frames (NF=1) only

68.4.9 VDOA IPU External Buffer 1 Frame 0 Address Register (VDOA_VDOAIEBA10)

Address: 21E_4000h base + 20h offset = 21E_4020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IEBA10																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

VDOA_VDOAIEBA10 field descriptions

Field	Description
IEBA10	External Address of Frame 0 output (IPU) buffer 1 - Note that the 3 LSB are always 0 (aligned to 8 address) Used in sync mode (SYNC=1) only

68.4.10 VDOA IPU External Buffer 1 Frame 1 Address Register (VDOA_VDOAIEBA11)

Address: 21E_4000h base + 24h offset = 21E_4024h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IEBA11																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

VDOA_VDOAIEBA11 field descriptions

Field	Description
IEBA11	External Address of Frame 1 output (IPU) buffer 1 - Note that the 3 LSB are always 0 (aligned to 8 address) This register is used only in sync mode (SYNC=1), 3 frames transfer (NF=1)

68.4.11 VDOA IPU External Buffer 1 Frame 2 Address Register (VDOA_VDOAIEBA12)

Address: 21E_4000h base + 28h offset = 21E_4028h

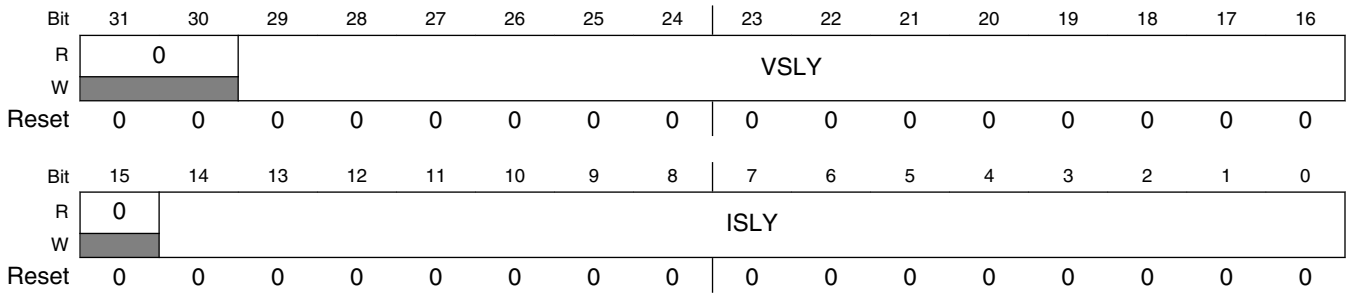
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IEBA12																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

VDOA_VDOAIEBA12 field descriptions

Field	Description
IEBA12	External Address of Frame 2 output (IPU) buffer 1 - Note that the 3 LSB are always 0 (aligned to 8 address) This register is used only in sync mode (SYNC=1), 3 frames transfer (NF=1)

68.4.12 VDOA IPU Stride Line Register (VDOA_VDOASL)

Address: 21E_4000h base + 2Ch offset = 21E_402Ch

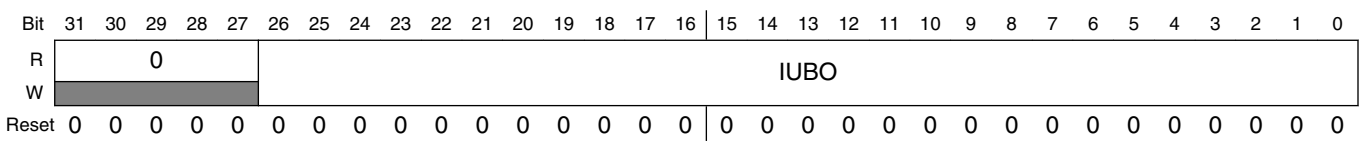


VDOA_VDOASL field descriptions

Field	Description
31–30 Reserved	This read-only field is reserved and always has the value 0.
29–16 VSLY	VPU Stride Line - Address vertical scaling factor in bytes for memory access. Also number of maximum bytes in the "Y" component row according to memory limitations.
15 Reserved	This read-only field is reserved and always has the value 0.
ISLY	IPU Stride Line - Address vertical scaling factor in bytes for memory access; also number of maximum bytes in the "Y" component row.- Note for 4:2:2 format ISLY will be doubled since each pixel consists of two bytes.

68.4.13 VDOA IPU U (Chroma) Buffer Offset Register (VDOA_VDOAIUBO)

Address: 21E_4000h base + 30h offset = 21E_4030h



VDOA_VDOAIUBO field descriptions

Field	Description
31–27 Reserved	This read-only field is reserved and always has the value 0.
IUBO	The offset of Chroma (UV) Buffer for all IPU output frames i.e Buffer Chroma address will be VDOAIEBAnm+VDOAIUBO - Note that the 3 LSB are always 0 (aligned to 8 address). This parameter is used only for PFL = 4:2:0

68.4.14 VDOA VPU External Buffer 0 Address Register (VDOA_VDOAVEBA0)

Address: 21E_4000h base + 34h offset = 21E_4034h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	VEBA0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

VDOA_VDOAVEBA0 field descriptions

Field	Description
VEBA0	Address of Frame 0 VPU buffers - Note that the 3 LSB are always 0 (aligned to 8 address) Used for all transfers

68.4.15 VDOA VPU External Buffer 1 Address Register (VDOA_VDOAVEBA1)

Address: 21E_4000h base + 38h offset = 21E_4038h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	VEBA1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

VDOA_VDOAVEBA1 field descriptions

Field	Description
VEBA1	Address of Frame 1 VPU buffers - Note that the 3 LSB are always 0 (aligned to 8 address) Used when transfering three frame (NF=1) only

68.4.16 VDOA VPU External Buffer 2 Address Register (VDOA_VDOAVEBA2)

Address: 21E_4000h base + 3Ch offset = 21E_403Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	VEBA2																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

VDOA_VDOAVEBA2 field descriptions

Field	Description
VEBA2	Address of Frame 2 VPU buffers - Note that the 3 LSB are always 0 (aligned to 8 address)

VDOA_VDOAVEBA2 field descriptions (continued)

Field	Description
	Used when transfering three frame (NF=1) only

68.4.17 VDOA VPU U (Chroma) Buffer Offset Register (VDOA_VDOAVUBO)

Address: 21E_4000h base + 40h offset = 21E_4040h

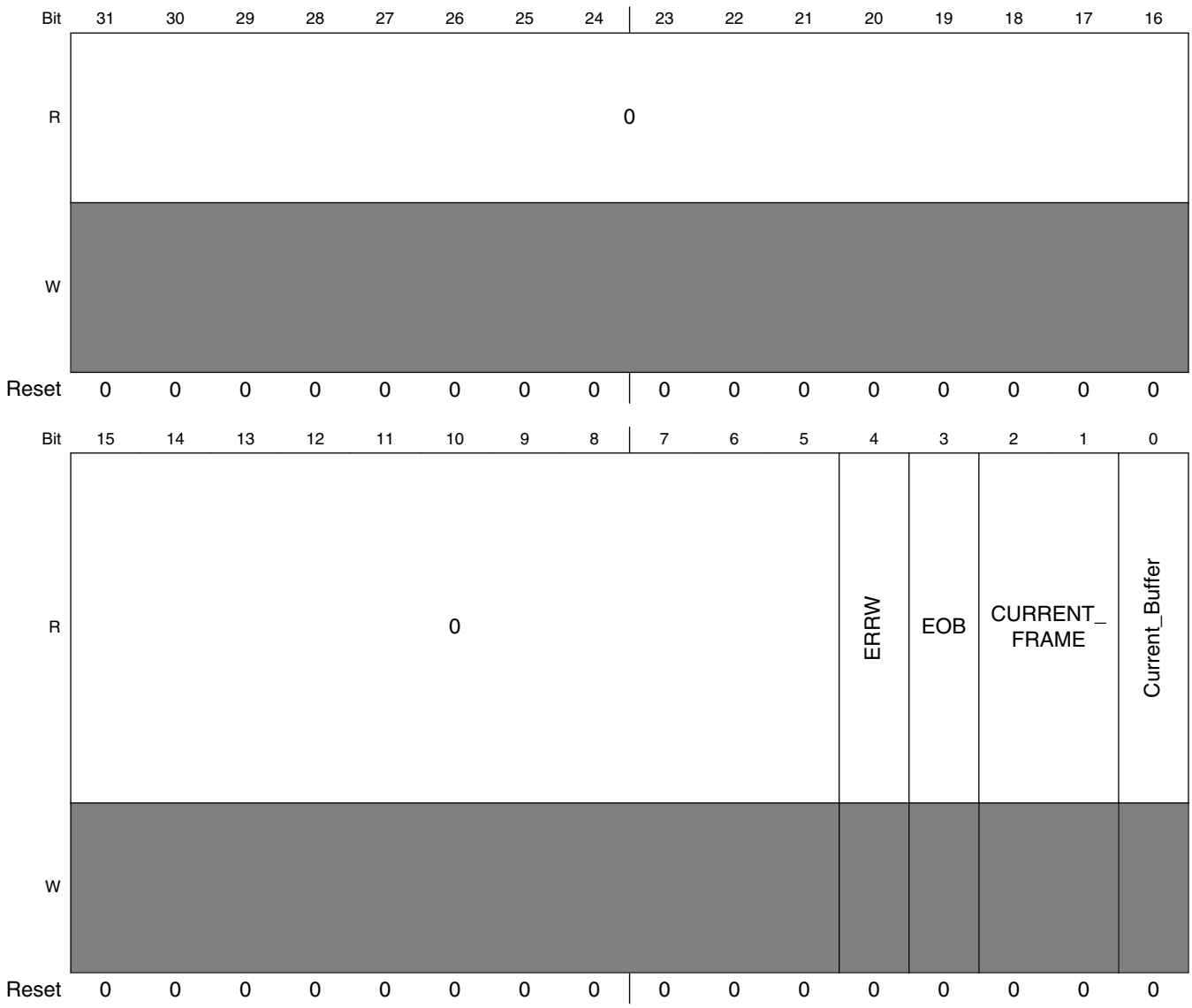
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0					VUBO																										
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

VDOA_VDOAVUBO field descriptions

Field	Description
31–27 Reserved	This read-only field is reserved and always has the value 0.
VUBO	The offset of Chroma (UV) Buffer for all VPU input frames i.e Chroma Buffer address will be VDOAVEBAm+VDOAVUBO - Note that the 3 LSB are always 0 (aligned to 8 address)

68.4.18 VDOA Status Register (VDOA_VDOASR)

Address: 21E_4000h base + 44h offset = 21E_4044h



VDOA_VDOASR field descriptions

Field	Description
31–5 Reserved	This read-only field is reserved and always has the value 0.
4 ERRW	Error Write - Indicates that the last access that failed was a read or a write.This field is valid only when TERR bit is set in VDOA Interrupt Status Register - VDOAIST 0 **READ_ERROR** — Read Error 1 **WRITE_ERROR** — Write Error

Table continues on the next page...

VDOA_VDOASR field descriptions (continued)

Field	Description
3 EOB	End of Band- Indicates that the VDOA has finished transferring a band in SYNC mode and is waiting for IPU to continue.
2–1 CURRENT_ FRAME	Current Frame - When working on 3 frames the index of the frame currently transferred
0 Current_Buffer	Current Buffer - for Double buffer shows the index of the buffer currently transferred

Chapter 69
Video Processing Unit (VPU)

69.1 Overview

Video Processing Unit of i.MX 6Dual/6Quad is a high performance multi-standard video codec

which can decode H.264 BP/CBP/MP/HP, VC-1 SP/MP/AP, MPEG-4 SP/ASP, H.263 P0/P3, MPEG-1/2 MP, Divx (Xvid) HP/PP/HTP/HDP, RV8/9/10, VP8 (1280x720), AVS, H.264-MVC (1280x720), MJPEG BP (max. 8192x8192) up to full-HD 1920x1088 @30fps plus D1 @30fps. It can also perform H.264 BP/CBP, MPEG-4 SP, H.263 P0/P3, MJPEG BP (max. 8192x8192) encoding up to full-HD 1920x1088@30fps. VPU can support encode or decode multiple video clips with multiple standards simultaneously.

NOTE

RealNetworks video codec is disabled by default on i.MX 6 series processors. Please contact your Freescale sales representative for more details.

VPU has two bus interface: IP bus for register access controlling and AXI bus for data throughput.

VPU makes use of external memory space for bitstream buffer, parameter buffer, bit code buffer, working buffer and frame buffer.

VPU has an embedded BIT processor controlling internal video processing sub-blocks and communicating with host processor through host interface. Very low resources of the ARM platform is required to support VPU. Host processor only need to access VPU registers for initializing VPU or setting parameters during frame gap. In application, it is done through VPU API called by host processor. Large part of video codec (except BIT processor) is optimally shared which enables to achieve low power and low gate count.

VPU combined with processing capabilities of Image Processing Unit (IPU) can support high quality video processing applications for the chip.

The following figure shows VPU top-level diagram.

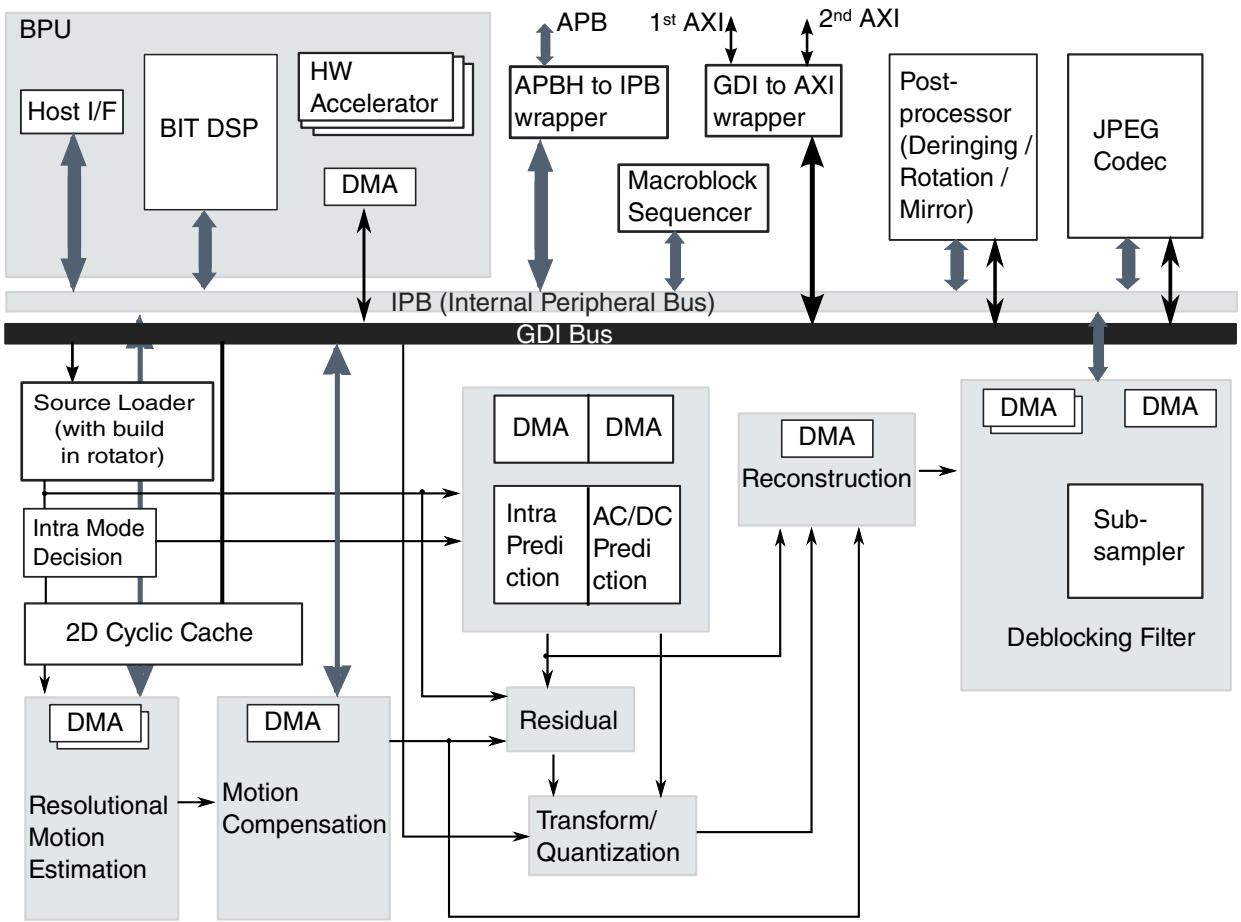


Figure 69-1. VPU Block Diagram

69.1.1 Features

VPU supports H.264 BP/CBP/MP/HP, VC-1 SP/MP/AP, MPEG-4 SP/ASP, H.263 P0/P3, MPEG-1/2 MP/HP, DivX (Xvid) HP/PP/HTP/HDP, RV8/9/10, VP8 (1280x720), AVS, H.264-MVC (1280x720), MJPEG BP decoding and H.264 BP/CBP, MPEG-4 SP, H.263 P0/P3, MJPEG BP encoding. The supported resolution is up to full-HD 1920x1088@30fps plus D1@30fps for decoding and full-HD 1920x1088@30fps for encoding except MJPEG codec (up to 8192x8192). VPU can support various error resilience tools and also support multiple decoding and full duplex multi-party-call simultaneously.

And the VPU provides programmability, flexibility and ease of upgrade in decoding/encoding or host interface because all the controls in decoding/encoding process and host interface are implemented as firmware in a programmable BIT processor.

NOTE

The capabilities of VPU listed here is measured with a nearly ideal bus and memory system. But in real SOC, the capabilities of VPU are also depends on the bus and memory system of SOC.

The detailed feature of VPU is as follows:

Table 69-1. Supported Decoding/Encoding Standards

	Standard	Profile	Level	Resolution
Encoder	H.264	BP/CBP	4.0	1920x1088
	MPEG-4	SP	5/6	1920x1088
	H.263	P0/P3	70	1920x1088
	MJPEG	BP		8192x8192
Decoder	H.264	BP/MP/HP	4.1	1920x1088
	MPEG-4	SP/ASP		1920x1088
	H.263	profile3	70	1920x1088
	VC-1	SP/MP/AP	3	1920x1088
	MPEG-2	MP/HP	High	1920x1088
	DivX/Xvid	HP/PP/HTP/HDP		1920x1088
	VP8			1280x720
	AVS	Jizhun	6.2	1920x1088
	RV	8/9/10		1920x1088
	H.264-MVC	BP/MP/HP		1280x720
	MJPEG	BP		8192x8192

- Multi-standard video codec
  - H.264/AVC encoder for baseline profile
  - MPEG-4 encoder for simple profile
  - H.263 encoder for baseline profile
  - MJPEG encoder for baseline profile
  - H.264/AVC decoder for baseline profile, constraint baseline profile, main profile and high profile
  - VC-1 decoder for simple profile, main profile and advanced profile
  - MPEG-4 decoder for simple profile, advanced simple profile except GMC
  - H.263 decoder for baseline profile + Annex I, J, K and T
  - DivX decoder for the Handheld/Portable/Home Theater/High Definition profiles
  - MPEG-2 decoder for main profile @ main and high level

    - RV decoder for profile 8/9/10
    - VP8 decoder fully compatible with On2 VP8 decoder specification
    - AVS decoder supports Jizhun profile 6.2
    - H.264 MVC decoder for baseline profile, main profile and high profile
    - MJPEG decoder for Baseline profile
- Other features
    - Pre/Post rotator and mirror
    - Built-in de-ringing filter.
    - Built-in de-blocking filter for MPEG-2/MPEG-4/DivX.
    - MPEG-2 encoder partial acceleration.
    - JPEG supports down-sampler with the down-sampling ratio of 2:1, 4:1, and 8:1 in either pic-width or pic-height
    - VPU supports arbitrary number (e.g., 16 or more) of decoding/encoding processes simultaneously. Each process can have a different format.
    - Color format: All codecs support only 4:2:0, except MJPEG codec which support 4:2:0, 4:2:2, 2:2:4, 4:4:4 and 4:0:0
    - Robust error detection/concealment.
- Optimal external memory accesses
    - Configurable frame buffer formats (linear to tiled) for longer burst-length
    - 2D cache for motion estimation and compensation to reduce external memory accesses
    - secondary AXI port for on-chip memory to enhance performance
- Programmability
    - Embeds 16-bit BIT processor that is dedicated to processing bitstream and controlling the hardware.
    - General purpose registers and interrupt generation for communication between host processor and VPU.

69.2 Modes of Operation

The VPU has two kinds of operating modes: normal operating mode and low power mode.

69.2.1 Normal Operating Mode

Normal operating mode is the video codec processing mode, which can be MPEG-4 encoding/decoding, DivX decoding, H.264 encoding/decoding, VC-1 decoding, or multiple bitstream encoding/decoding in multiple standards simultaneously.

VPU supports arbitrary number (e.g., 16 or more) of decoding/encoding processes simultaneously. Each process can have a different format. The host processor creates and executes the specified process by sending parameters and commands to the BIT processor through the VPU API.

69.2.2 Low Power Mode

When VPU is in idle state, VPU interface signal vpu_idle is asserted and system can gate off its clock source according to this signal.

If the clock source is gated off and system wants to wake it up, it just re-enable the clock and send command to VPU. If power is off when the system wants to wake it up, VPU has to be re-initialized. Refer to section Detailed Signal Descriptions for detailed description about vpu_idle signal.

69.3 External Signal Description

The VPU has no external signals connecting off-chip.

69.3.1 Detailed Signal Descriptions

All VPU input clock and reset sources come from SOC CCM. VPU also has internal multi-level clock gating scheme which can gate off VPU internal clocks to the unused sub-blocks.

VPU uses two 64-bit AMBA3 AXI bus interface for data transfer from/to external memory and internal OCRAM. VPU gasket is responsible for transferring AMBA APB bus to IP bus protocol.

The interrupt signal ipi_int_vpu, jpg_int_vpuare VPU interrupt signals, active high. When VPU interrupt is enabled in system, its corresponding interrupt service routine is called when ipi_int_vpu or jpg_int_vpu signal is raised. This signal is retained until host processor clear it. This signal is synchronized to the positive edge of the cclk.

The vpu_idle signal is used by CCM to control VPU Clock gating, active high. It indicates whether VPU is busy or idle. If vpu_idle is high, VPU is not running and system can cut off the clock source. If VPU clock is off, to wake up VPU, just turn on the clock and run VPU through BIT processor command.

69.3.2 New features related to System Level interface

i.MX 6Dual/6Quad VPU have 3 new features related to system level interface: memory protection, 64-byte burst writing back and sub-frame synchronization.

These features can avoid VPU writings to unexpected address and enhance performance.

69.3.2.1 Memory Protection

The VPU supports a memory write protection feature. This feature is designed to block off data write to an unexpected address.

There are region register sets which indicate writable address ranges. If there happens an AXI write request out of the accessible address range, this Write Protect Module will block the request.

69.3.2.2 64-Byte Burst Writing Back

Burst Write Back Module is designed to increase the burst size of write transactions.

The output pattern is an macroblock shape (or a similar rectangular shape) in decoding process. Luminance is 16x16 pixels and chrominance is 8x8 (or 8x16 if interleaved) pixels per macroblock. The macroblock shape output generates many short burst accesses in linear map. 16x16 block write requires sixteen times of 2 bursts, and 8x8 block write is eight times of 1 burst. These short bursts are not efficient in SDRAM access. In order to increase the burst size in writing,

Burst Write Back Module does buffering small size of blocks and writes them with burst size 8 (burst size 8 is 64 byte assuming the bus is 64 bits).

The frame index to be buffered is programmable. Host (or firmware) can set a frame index to apply burst write on a frame basis.

Burst Write Back Module can be used only to the frame index whose map type is a linear map.

69.3.2.3 Sub-frame Synchronization

VPU supports sub-frame synchronization by receiving dedicated sub-frame-ready signals from IPU (Image Process Unit) so that VPU encoder can start encoding process as early as the minimum set of raw video data is ready.

Typically VPU had to wait until one raw image is completely written into the frame buffer. But sub-frame synchronization makes VPU start encoding after a line of macroblock row buffer is filled up by IPU, and if there are multiple raw image buffers, IPU can write raw image to one buffer while VPU is encoding with other buffer, so that the latency time caused by image buffering can be significantly eliminated.

69.4 Clocks

There are four clock domains in VPU.

The following table describes the clock sources for VPU.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 69-2. VPU Clocks

Clock name	Clock Root	Description
aclk	vpu_axi_clk_root	AXI bus clock domain. Controls all AXI bus related functions. Maximum frequency is 266 MHz @ TSMC 40nm process. The aclk derives from the same clock as other blocks performing AXI bus access in the chip, the frequency is determined by the whole system performance requirement.
cclk	vpu_axi_clk_root	VPU functional clock. Controls all VPU decoding functionality, with a maximum frequency of 264MHz or 352 MHz depending on the voltage of VDD_PU_CAP. See the Operating Ranges table in the datasheet for details.
ipg_clk_s	ipg_clk_root	IP bus clock domain. Controls VPU registers read/write function, with a maximum frequency of 66.5 MHz. ipg_clk_s is a gated clock of IP green-line clock (ipg_clk) with ips_module_en, it is turned off when there is no registers read/write, for power saving.
rclk	video_27m_clk_root	Reference clock domain. Used as reference clock for vpu_idle related logic.

VPU has all four clock domains because it involves in AXI signal generation, functional calculation, IP bus configuration, vpu_idle control logic. Only positive edge clocks is used in VPU design. Each clock is fully asynchronous with other clocks and separated from other clocks.

All clocks can be gated off when VPU is in the idle state to reduce power consumption. There is no restriction about the value of aclk and cclk. The actual frequency of aclk and cclk affects bus bandwidth and video decoding processing capability.

69.5 Functional Description

VPU is a high-performance multi-standard video processing block that supports up to full HD 1920 x 1088 at 30 fps plus D1 at 30 fps decoding and 1920 x 1088 at 30 fps encoding.

69.5.1 VPU Architecture

VPU is a high performance multi-standard video codec that can decode

- H.264 BP/CBP/MP/HP
- VC-1 SP/MP/AP
- MPEG-4 SP/ASP
- H.263 P0/P3
- MPEG-1/2 MP/HP
- Divx (Xvid) HP/PP/HTP/HDP
- RV8/9/10
- VP8 (1280x720)
- AVS
- H.264-MVC (1280x720)
- MJPEG BP (max. 8192x8192) up to full-HD 1920 x 1088 at 30 fps plus D1 at 30 fps

It can also perform H.264 BP/CBP, MPEG-4 SP, H.263 P0/P3, and MJPEG BP (max. 8192x8192) encoding up to full-HD 1920 x 1088 at 30 fps. VPU can support encode or decode multiple video clips with multiple standards simultaneously.

It connects with the system via the 32-bit IP bus for system control and 64-bit AMBA3 AXI for data throughput. It uses on-chip memories to achieve high performance.

VPU has a 16-bit DSP called a BIT processor. The BIT processor controls the internal video codec sub blocks and communicates with a host processor through the host interface. The required resource for the host ARM platform to control the VPU is very low, under 1 MIPS, because all functions such as rate control, FMO, ASO, video codec control and error resilience are implemented in the BIT processor. The majority of sub-blocks in VPU are optimally shared, which enables the ultra low power and low gate count.

VPU mainly consists of an embedded 16-bit BIT processor, video codec hardware, and bus arbiter/interface. Refer to Figure 1 for the block diagram of VPU.

69.5.1.1 Embedded BIT processor

The embedded BIT processor is used for parsing or forming bitstreams. It includes some hardware accelerators to speed up bitstream processing.

In addition to handling bitstreams, the BIT processor controls the video decoding hardware and communicates with host processor through IP bus and AXI bus interface.

69.5.1.2 Video CODEC Hardware

All video decoding processing, except handling coefficients for VLD, are implemented in hardware.

The video codec hardware is designed to reduce logic gate count by sharing parts of sub-blocks for multi-standard video decoding.

It mainly includes the following hardware components.

69.5.1.2.1 Inter-Predictor

The Inter-Predictor sub-block uses a reconstructed motion vector that represents the displacement between the block currently being decoded and the corresponding location in the reference frame, to calculate interpolated pixel data for motion compensation.

69.5.1.2.2 AC/DC and Intra-Predictor

There are two intra-prediction sub-blocks in the video decoding hardware. One is for the MPEG-4/H.263 P3/VC-1 AC/DC prediction and another for the H.264 intra-prediction.

In case of VC-1 decoding, the hardware prediction mode decision is used. It brings high performance and low power consumption. The coefficient data of decoding is re-ordered automatically based on the detected prediction mode. For H.264 intra-prediction mode in decoding, hardware mode decision or software based made decision is used.

69.5.1.2.3 Inverse transform/Inverse quantization

VPU has only one transform/quantization sub-block. It operates in several types of inverse transform/quantization for MPEG-2, VC-1 and H.264.

Each inverse transform/quantization for decoding uses different coefficients and different data processes, but it uses same control and data flow in many cases.

69.5.1.2.4 De-blocking/Overlap-smoothing filter

>De-blocking filter removes blocking artifacts resulted from quantization, different motion vectors. The filter processing is applied in both decoder and encoder.

The VPU de-blocking filter supports H.264/H.263/MPEG4/VC-1. Overlap-smoothing filter supports VC-1 overlap-smoothing feature. For H.264, H.263 and VC-1, the de-blocking filter operates within coding loop. Filtered frames are used as reference frames for motion compensation of subsequent coded frames. But for MPEG4, the de-blocking filter operates outside coding loop for only display.

69.5.1.2.5 Coefficient buffer interface

The coefficient buffer interface provides a channel for BIT processor to read quantized coefficients resulted from encoding process or to send variable-length-decoded coefficients to the video codec for decoding process.

69.5.1.2.6 Macroblock controller

VPU has a complex and large number of pipeline for high-performance. To manage it wholly by BIT processor is not suitable, so VPU embeds the macroblock controller to control all sub-blocks of the video codec based on the configuration of pipelining by BIT processor. This scheme reduces the load on BIT processor and guarantees the programmability of the block. Before the video codec encodes or decodes a macroblock, BIT processor configures how the pipeline of the codec is structured. If all processes are completed for encoding/decoding a macroblock, the macro-block controller indicates its completion.

In summary, BIT processor configures which sub-blocks are enabled for current macroblock processing, and the macroblock controller manages corresponding sub-blocks based on the configuration.

The figure below shows the connectivity of macroblock controller.

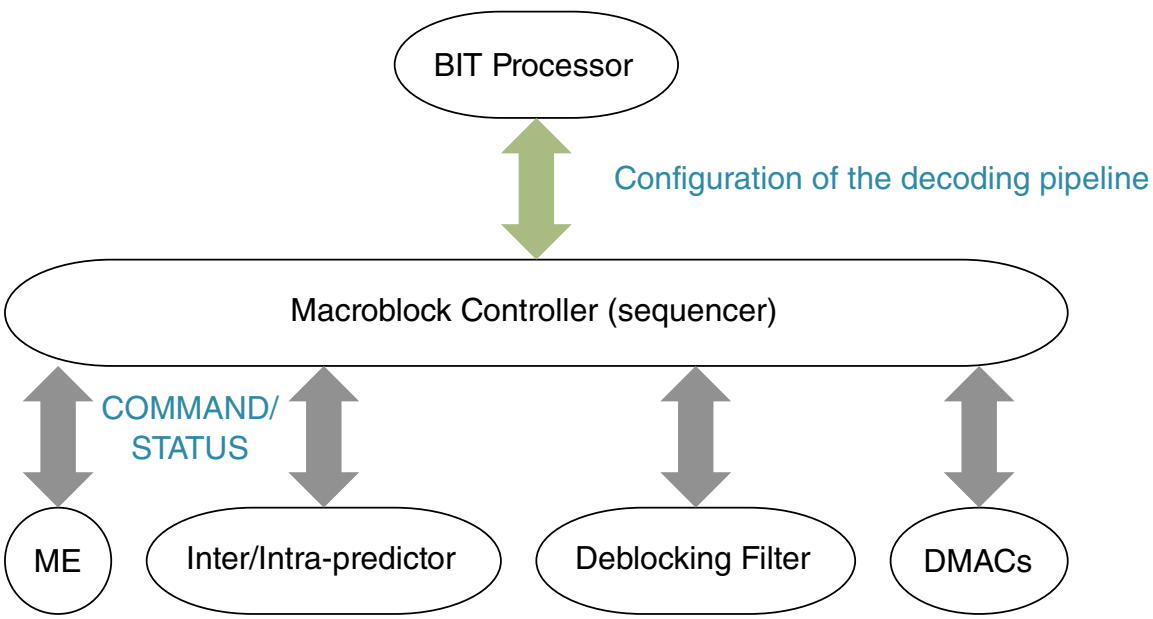


Figure 69-2. The connectivity of macroblock controller

69.5.1.2.7 Rotation/Mirroring

VPU supports rotation together with mirroring function for decoding output image to display. It is done by rotator/mirror sub-block.

The rotation/mirroring process in decoding process requires additional bandwidth because VPU has to re-use the un-rotated image for decoding the next image. So the rotated image is written to other memory space. In this scheme, the display I/F has not to change memory space for displaying the decoded image because subsequent rotated image is written to the same space.

The rotator sub-blocks support 8-types mode of 90 x n degree(n=0, 1, 2, 3) rotating and mirroring. The figure below gives architecture diagram of post rotation/mirroring sub-block.

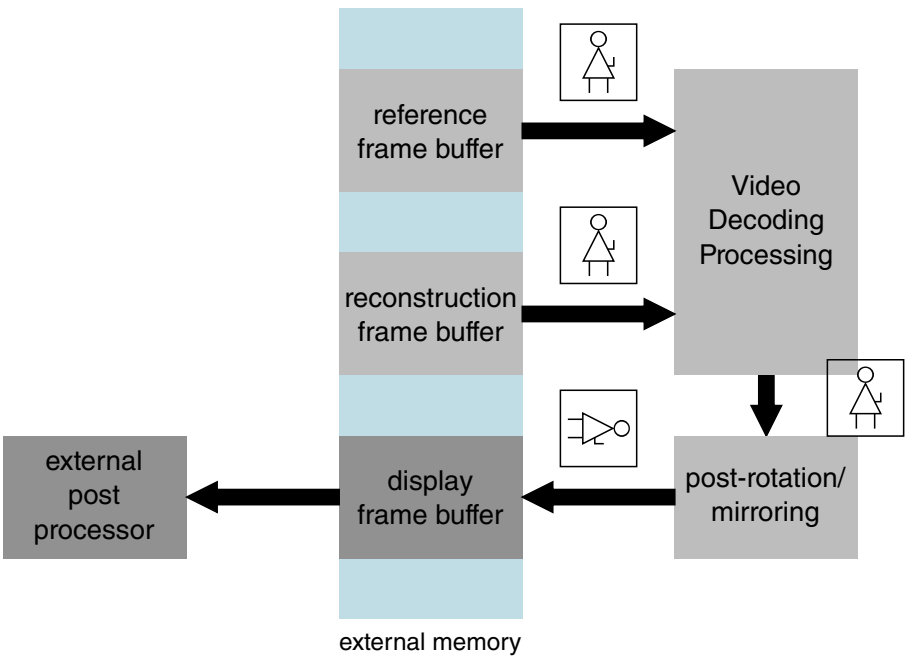


Figure 69-3. Post rotation/mirroring

69.5.1.2.8 Cache

In VPU a cache hardware block is used for reduction of data transactions on external memory bus.

Most of all transaction come from/to frame buffers which store pixels in 2-D coordinates.

The cache buffer consists of 2-D cache blocks which store 2mx2n pixels.

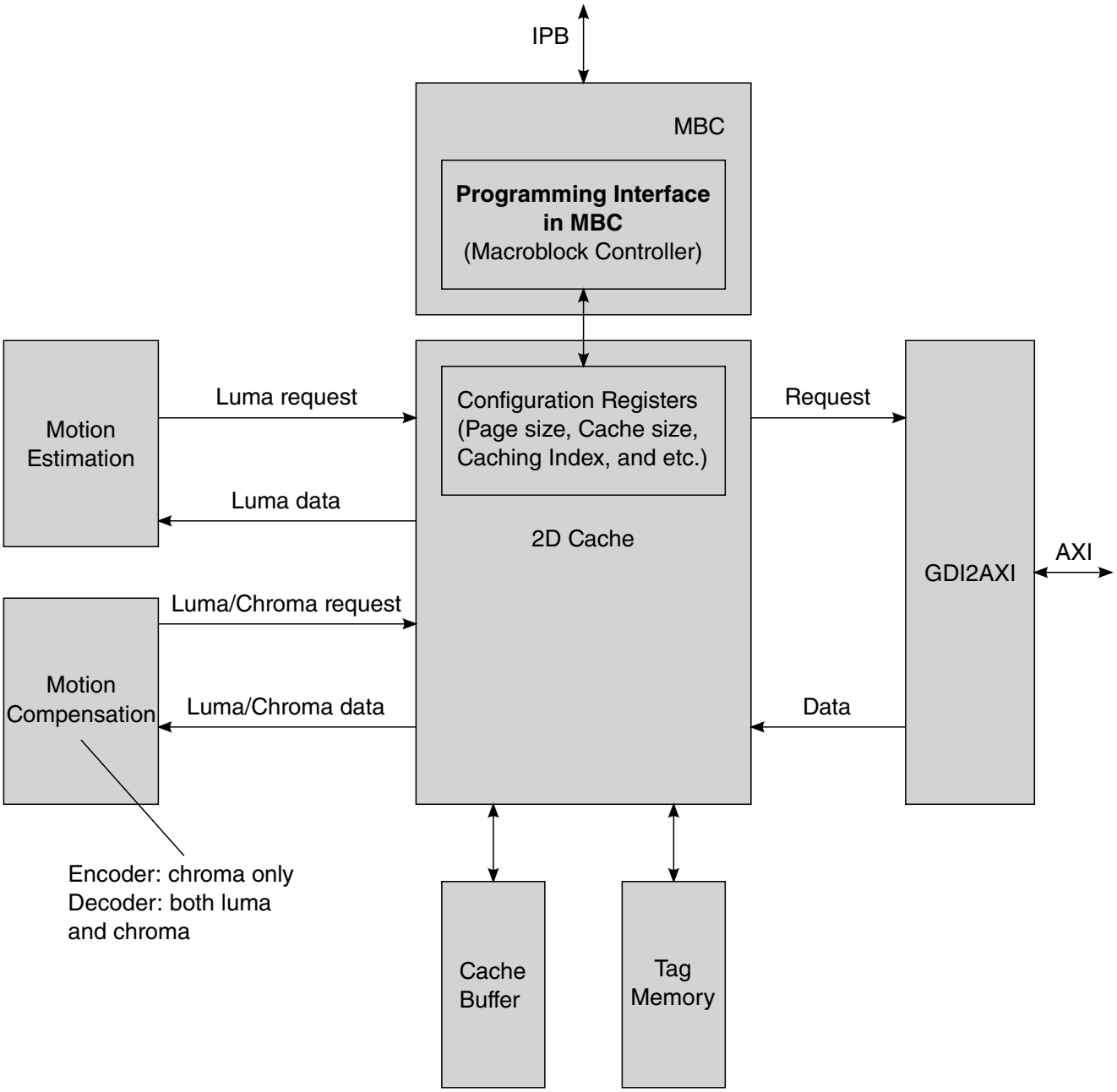


Figure 69-4. Cache Hardware Block

It shows the best performance especially in conjunction with tiled format. In case of linear format, pure bandwidth can be reduced using 2-D cache, but the burst-length, one of critical factors affecting bus efficiency in system, typically tends to be very short. But, in use of cache in tiled format its typical burstlengths are 32-(for chroma pixels) or 64-bytes(for luma pixels). The cache buffer size is pre-configurable, which means the size is determined at RTL compile time. Within the given cache buffer size, the size of cache block and total number of cache blocks are programmable in runtime.

Features

- Direct mapped cache with split index
- Programmable cache block size
  - A cache block can store 2-D block of 2m x2n
- Configurable cache size
  - Total number of cache blocks = 2ix2j (i and j are programmable, but the number is restricted to the practical memory size for the cache.)
- Caching specified single or two reference frame(s)
- The default cache block size is 8x8, and total number of cache blocks is 64 cache blocks.(horizontal 8 x vertical 8 cache blocks)

69.5.2 Reset

There are four reset signals going into VPU, corresponding to four clock domains, active low.

- areset_n: used in AXI bus interface. Corresponding clock is aclk.
- creset_n: used in BIT processor and video codec hardware. Corresponding clock is cclk.
- ipg_hard_pos_async_reset_b: used in IP bus interface. Corresponding clock is ipg_clk_s.
- rreset_n: used in reference counter for vpu_idle or vpu_underrun. Corresponding clock is rclk.

The number of cycles for each reset signal must be at least 16 cycles.

VPU embeds an internal reset controller for feature of the software reset from the BIT processor. If any of the VPU blocks except the BIT processor is needed to reset (software reset), the host processor can enable this software reset by setting the software reset register through VPU API. But, the BIT processor cannot be reset by this software reset scheme, because the reset signal of the BIT processor is connected directly from external reset signal.

If reset fires when VPU is processing a transaction through the AXI bus, there is no guarantee that the AXI will complete the transaction normally, because VPU will be reset. If there are any corrupted data in memory, it can be discarded by software. Basically, if the host processor needs to issue a reset, it must check that there is no transaction on AXI bus between the VPU and external memory interface. In general, the AXI bus is free of VPU transactions when one frame decoding is completed. The start of next frame processing need software configuration.

69.5.3 Interrupts

There aretwo interrupt signals output from VPU. Basically, these interrupts are used to indicate completion of decoding or encoding one frame.

ipi_vpu_int_jpg is used to indicate complition of decoding or encoding one frame in MJPEG codec. ipi_vpu_int is used for other standards.They are generated when VPU interrupt is enabled and as well as the interrupt condition is met. The interrupt signal is active high and retains untill the host processor clears it by writing "1" to the interrupt clear register. This signal is synchronized to the positive edge of cclk.

When getting frame completion interrupt, the software can set the parameters for next frame processing and start the BIT processor again. The parameters are mainly the source/destination frame buffer base address. It can be different from the previous frame buffer because the previous completed frame of data may be needed for other image block like display block or IPU in the system.

Basically, the following operation responding to interrupt is dependent upon the application. For example, the software can send the decoded frame to the Image Processing Unit for post-processing and display, at the same time the software should prepare the next bitstream to be processed to the external memory before starting a new processing. This can be done through VPU driver.

69.5.4 Endianness

VPU supports both little and big endian memory system.

User should specify the endianness of the bitstream buffer and frame buffer corresponding to the application scenario. The endianness configuration can be done through VPU API.

69.6 Initialization Information

VPU embedded BIT processor is highly optimized to handle bitstream data.

In addition to processing bitstreams, the BIT processor also controls the communication between VPU and host processor.

The BIT processor firmware to drive video decoding is divided into 2 parts.

One is boot code, which is downloaded by host processor through the IP bus to BIT processor internal memory, it is loaded only once at the VPU initialization stage. The size of the boot code is 1Kbyte. This boot code has also to be written to a region of external memory, and the base address of the firmware region is written through VPU API.

The other is a package of firmware for driving decoding processing. Before starting decoding, firmware has to be written to the continuous region of external memory following the boot code. At run-time, the BIT processor will self-download firmware into internal memory corresponding to the activated decoding standard. It is loaded through the AXI bus.

The VPU initialization process is shown in the figure below.

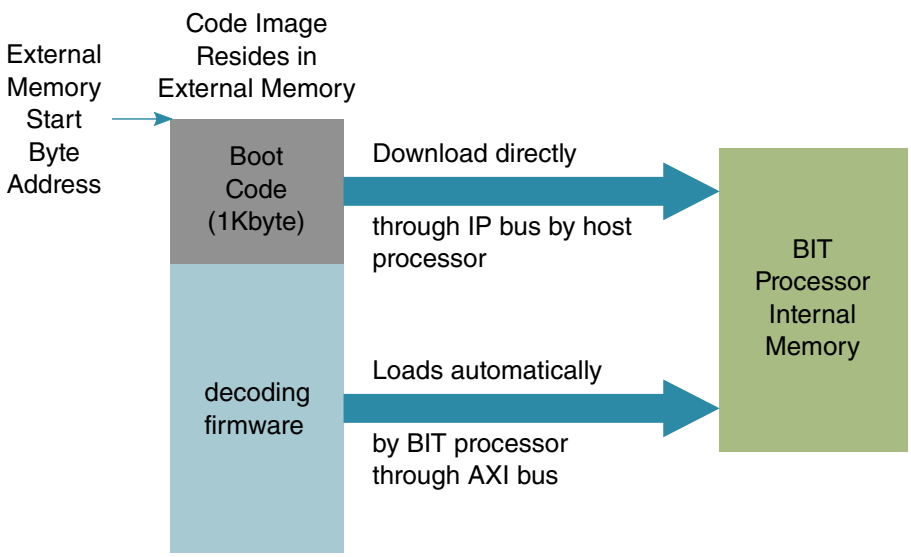


Figure 69-5. VPU initialization process

69.7 Application Information

The figure found here shows roles of the BIT processor and VPU video processing core sub-block and how to interface with application software.

Basically, at the frame level, host processor communicates with VPU through the provided API's. To give the video decoding more flexibility and debugging capability, all processes related to the bitstream are assigned to the BIT processor.

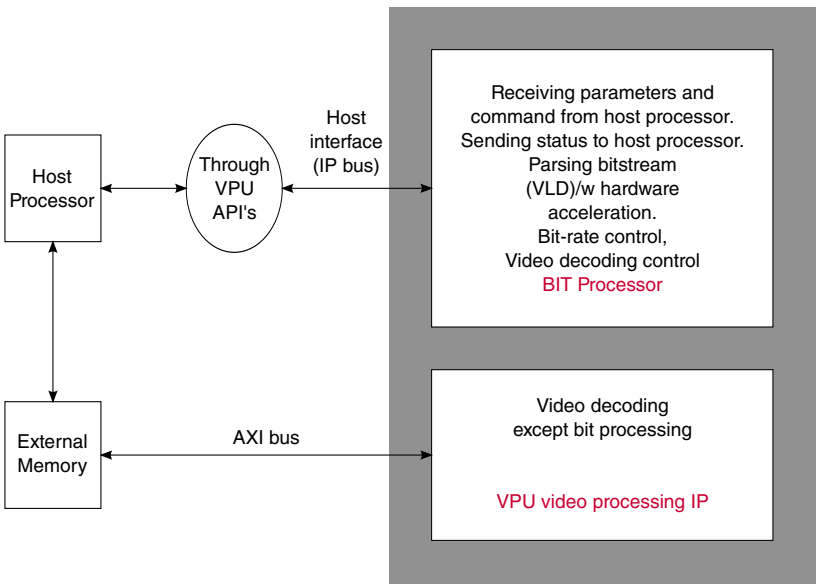


Figure 69-6. VPU interface with application software diagram

69.7.1 Video Decoding Processing Control

This section describes how BIT processor controls the video decoding and communicates with the host.

69.7.1.1 Video Decoding Process Flow

VPU can handle arbitrary number (e.g., 16 or more) of decoding/encoding processes simultaneously. Each process can have a different format - MPEG-4, MPEG-2, H.264, VC-1 and etc. The figure below shows a simplified state diagram for running decoding process.

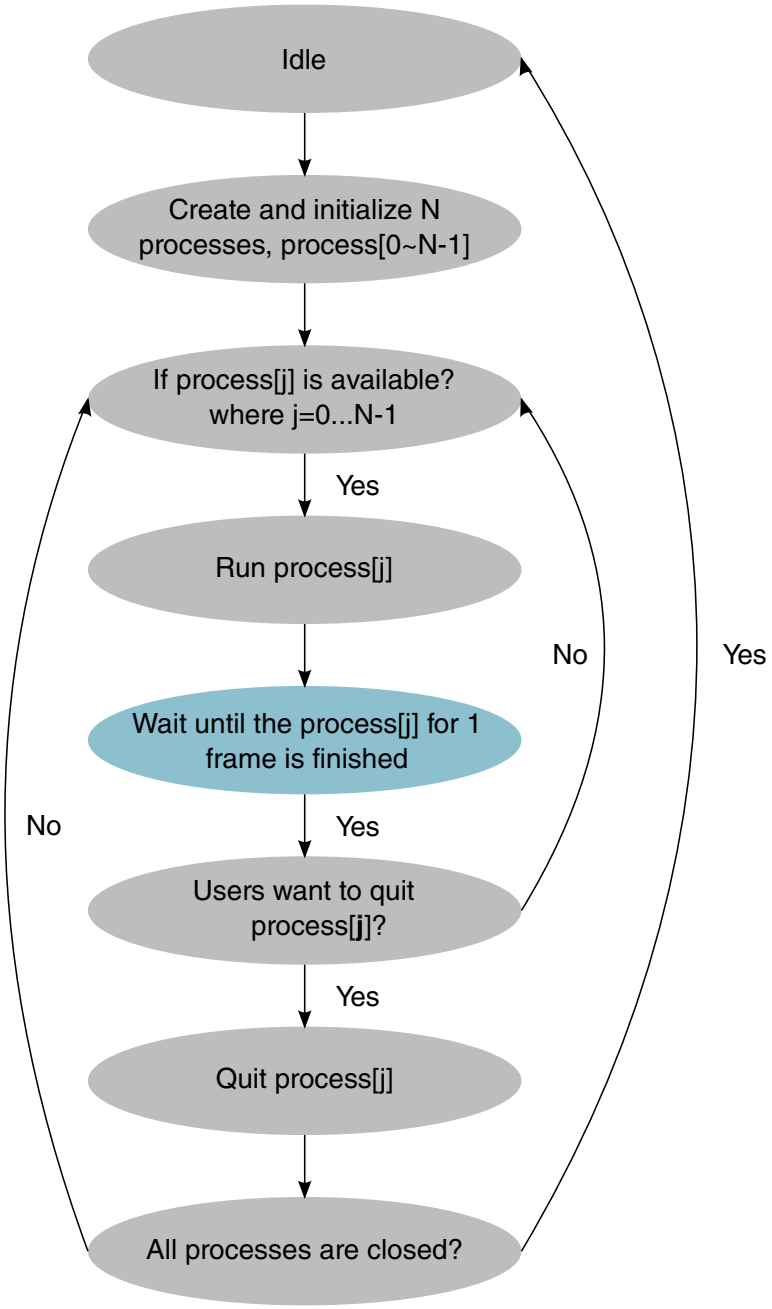


Figure 69-7. Decoding process state diagram

Each decoding process consists of three categories:

- Create processes- Software creates and configures processes.
- Running processes- At a proper time instance, software will begin a specific process. The proper time instance means when the decoding is in idle state and bitstream to be decoded is ready in the external memory.
- Quit Processes- Software can quit a specific process

If more than one process are ready to run, each process must be assigned to different process ID - RunIndex, which is range from 0 to 3. Basically, the ID is assigned based on the order of creation. For example, when 1 MPEG-4 Decoding + 1 H.264 Decoding + 1 MPEG-2 Decoding + 1 VC1 Decoding are running simultaneously, MPEG-4 Decoding is assigned to process index "0", H.264 Decoding is assigned to process index "1", MPEG-2 Decoding is assigned to index "2", and VC1 decoding is assigned to process index "3".

There is no priority rules for executing processes, after creating all processes at the initialization step, the host enables the BIT processor to execute process specified with the RunIndex. All processes are executed in time-division like mechanism, after one process finishes decoding or encoding a frame, another process then can be executed.

In conjunction with the process ID, the RunCodStd needs to be set, to define which coding standard is used with the created process and whether the created process will decode a bitstream. The table below shows the dedicated RunCodStd value for each decoding standard. All this can be done through VPU API.

Table 69-3. RuncCodStd Register Value for Coding Standard

Coding standard	RunCodStd
H.264 decoding	0
VC-1 decoding	1
MPEG-2 decoding	2
MPEG-4/DivX encoding	3
RV-8/9/10 decoding	4
AVS decoding	5
MJPEG decoding	6

69.7.1.2 Video Decoding Process Command

There are 7 execution commands to initialize, run, quit, set frame and set parameter processes. A command is sent by writing the command value to the RunCommand register through VPU API.

- DEC_SEQ_INIT- This command is to initiate a decoding process. At this command, BIT processor finds sequence header and parses the header to extract bitstream information such as picture size then the information is reported to DEC_SEQ_INIT return registers. API should set following configuration parameters before sending this command to VPU.
  - Bitstream buffer base address and size
  - Frame buffers base addresses and stride lines
- DEC_SEQ_END- This command is to terminate a decoding process.

    - After this command, no more PICTURE_RUN commands will be accepted for this process.
- DEC_PIC_RUN- This command is to decode one picture.
    - In decoding case, frame destination address should be set before executing.
- SET_FRAME_BUF- This command informs frame buffer addresses to be used as a decoding/reconstructing image to the BIT processor. Total 63 frame buffers may be used for decoding/reconstructing.
    - The decoding image must be reserved for motion compensation reference until it is not used for reference. So the decoded frame buffer is re-used carefully. The BIT processor receives the whole frame buffer address by this command before picture decoding is started. Then the BIT processor manages the frame buffer allocation for next storage area for decoding.
    - The frame buffer addresses are stored to external memory address. The luminance and one/two chrominance buffer addresses for each frame index must be stored.
- DEC_PARA_SET- This command adds a sequence parameter set or a picture parameter set in H.264 decoder case.
    - The sequence parameter set or the picture parameter set may be conveyed via "out-of-band". In that case, the host must transfer the sequence parameter set or picture parameter set to decoder by this command.
    - The sequence parameter set or picture parameter set must be written to the Parameter buffer of the BIT processor in RBSP format prior to executing this command. The BIT processor decodes the transferred sequence/the picture parameter and stores decoded contents. The decoded sequence/picture parameter set will be activated at decoding slice header with the matched sequence/picture parameter set id.
    - Multiple sequence/picture parameter sets may be delivered to decoder. They are distinguished by different sequence/picture parameter set id. BIT processor can process 32 sequence parameter sets and 256 picture parameter sets.
    - The type (sequence or picture) and size (byte count) of conveyed sequence/ picture parameter set must be delivered to BIT processor by command argument register.
- DEC BUF FLUSH- Flush data in bitstream buffer.
    - After this command finished, bitstream buffer read pointer will be 0. So host must set bitstream buffer write pointer as 0.
- GET F/W VERSION- Get firmware version.

There is a busy status register in the VPU to show whether the BIT processor is executing a command. The busy status keeps "1" until the command is finished, then the BIT processor can accept a new command.

69.7.1.3 Video Decoding Process Finish Detection

VPU raises the interrupt signal or busy state is asserted when a process finishes decoding a frame. So there are three ways of detecting whether the process is finished:

- Polling VPU interrupt status register. When interrupt is generated, the interrupt status register can indicate that.
- Polling VPU busy status register. When a DEC_PIC_RUN command is issued to BIT processor and the process is under operation, the busy status keeps "1", as soon as the busy status becomes "0", the decoding process is finished.
- Capture the interrupt signal from system level, respond to interrupt request within system interrupt service routine.

There is a single busy status register for all processes, user knows which process is finished by the specified running process ID - RunIndex. Interrupt status can be cleared by writing 1 to interrupt clear register. Busy state can be self cleared after read out.

69.7.1.4 Video Decoding Process Flow Example

The figure below shows a process flow example for decoding an H.264 bitstream and decoding an MPEG4 bitstream simultaneously. At first both decoding process are created and initialized, then each process is executed with DEC_PIC_RUN command alternately. More details are described as below.

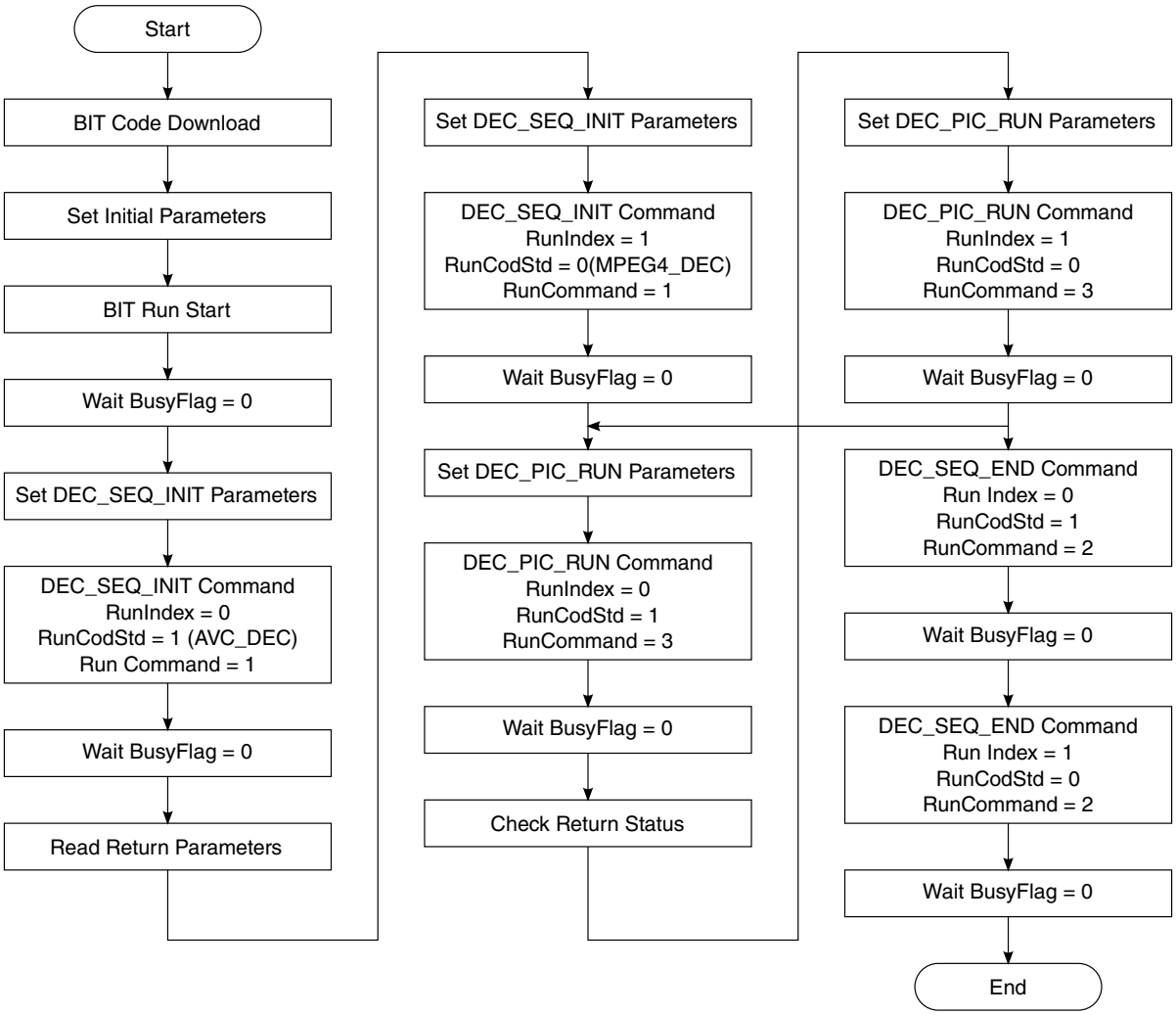


Figure 69-8. H.264 decoding and MPEG4 decoding process flow

NOTE

*RunCommand = 1 (DEC_SEQ_INIT); RunCommand = 2 (DEC_SEQ_END); RunCommand = 3 (DEC_PIC_RUN)

1. Initialize VPU
   - BIT Code Download: Load BIT Processor firmware to memory.
   - Set Initial Parameters: General configuration for BIT processor, setting working buffer base address, BIT Code memory address, bitstream buffer control and so on.
   - BIT Run Start: Run BIT processor to initialize VPU.
2. Create and initialize an H.264 decoding process
   - Set DEC_SEQ_INIT parameters: Configure base address and size of bitstream buffer, base address of frame buffers and so on.
   - Run DEC_SEQ_INIT command: Initiate an H.264 decoding process.

- Wait BusyFlag=0: Wait BIT processor completes DEC_SEQ_INIT command execution.
- Read Return Parameters: Read the features of decoded bitstream, such as the picture resolution and number of reference frames, this can be done by reading the RetSrcFormat and Ret264Info register through VPU API. In this way, the host can prepare the required frame buffers.

3. Create and initialize an MPEG4 decoding process
   - The flow is similar to the H.264 decoding process except the run standard setting.
4. Run the H.264 decoding process
   - Set DEC_PIC_RUN Parameters: Configure the frame destination address.
   - Run DEC_PIC_RUN command: Start the H.264 decoding process.
   - Wait BusyFlag=0: Wait the BIT processor completes DEC_PIC_RUN command execution. It also means one frame process is finished. (The finish detection can be implemented in other way described in Video Decoding Process Finish Detection.) The decoded frame can be sent to the Image Processing Unit for post-processing and display. The actual operation is dependant on the application.
5. Run the MPEG4 decoding process
   - The flow is similar to the H.264 decoding process. The decoding process should configure frame source address in addition to destination address.
6. Execute step 4 and step 5 alternately.
   - Before running decoding process, the host should load new bitstream to the bitstream buffer if it is empty, and update the frame destination address according to the application.
7. Stop the decoding processes
   - Run DEC_SEQ_END command to each process to terminate it.

Basically, the process flow for decoding is similar, though it may have minor change for different firmware version. The detailed implementation is done in the VPU driver.

69.7.2 Video Encoding Processing Control

This section describes how BIT processor controls the video encoding and communicates with the host.

69.7.2.1 The Pipeline for Encoding

The following figure shows the 11 pipeline stages for H.264 encoding.

In encoding, there are two separate data paths. The nine stages on the right form the encoding path, and the four stages on the left form the reconstruction data path.

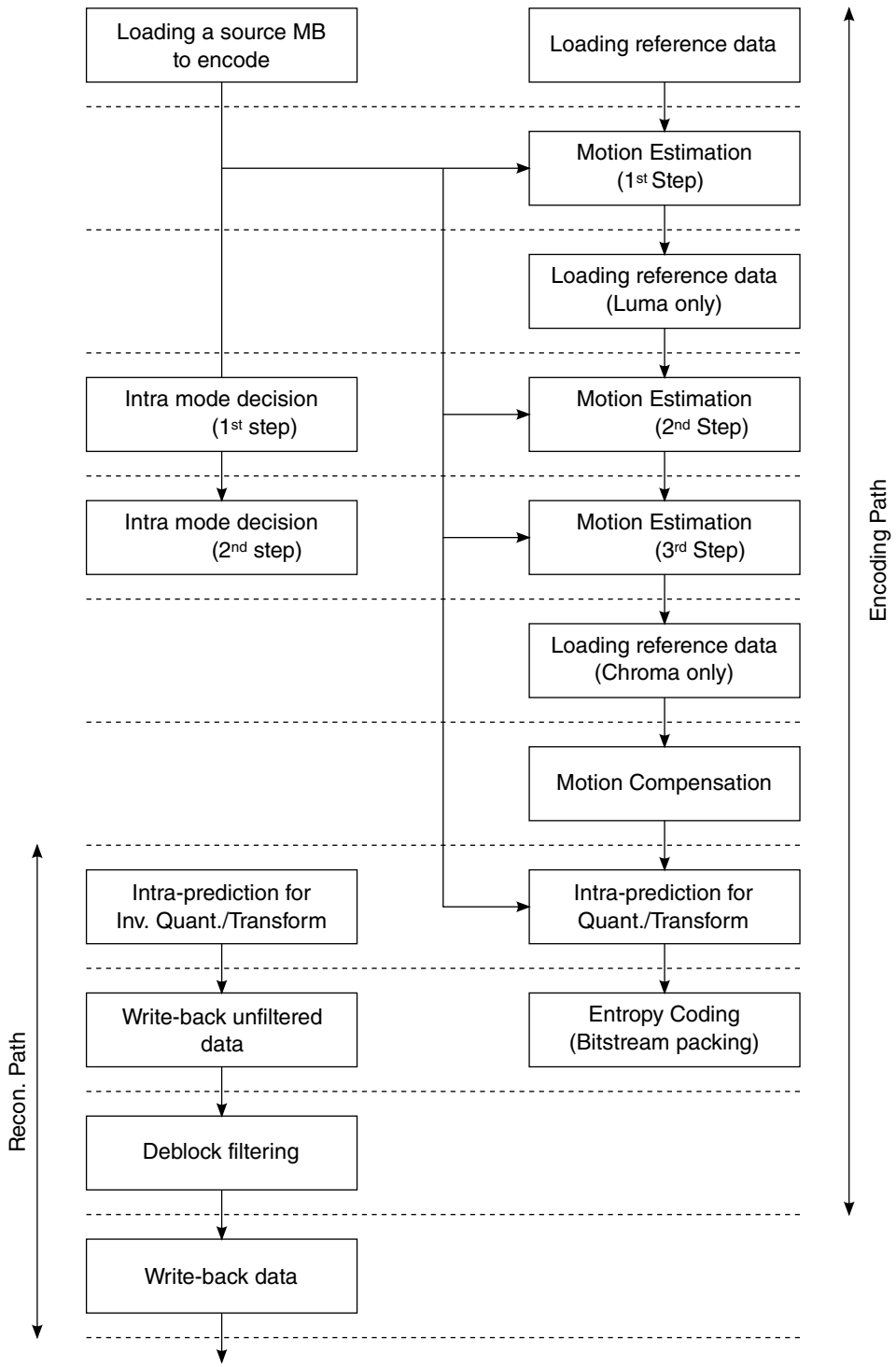


Figure 69-9. Encoding pipeline stage for H.264

The sub-block in each stage is controlled by BIT processor and main controller (macroblock sequencer). The detailed H.264 encoding pipeline stage is as follows.

- Pipeline Stage 1
  a. Load source and reference data through a sub-sampler.
  b. Loaded data passes through a sub-sampler.
- Pipeline Stage 2
  a. The motion estimation (ME) searches coarse motion vectors.
- Pipeline Stage 3
  a. Load data for refine motion estimation.
- Pipeline Stage 4
  a. Excute refine ME and fast intra-mode decision
- Pipeline Stage 5
  a. Excute fractional ME and fine intra-mode decision.
- Pipeline Stage 6
  a. Load reconstructed data for motion compensation.
- Pipeline Stage 7
  a. The MC block writes predicted data for the ME motion vector.
  b. The intra predictor calculates Intra SAD and cost for mode decision.
  c. The Inter/Intra prediction block writes predicted data local memory for subtraction in the next pipleline stage.
- Pipeline Stage 8
  a. Execute Intra Prediction and transfer the result to Quantizer/Transform.
  b. The quantized and transformed data are written to residual buffer for bitstream packing.
  c. The quantized and transformed data are transferred to the Inverse Quantizer/ Inverse Transfrom.
  d. The inverse quantized and inverse transformed data are added(reconstructed) with prediction block data, to make reconstructed frame.
- Pipeline Stage 9
  a. The BIT processor reads the residual buffer and makes the bitstream.
  b. Write back the un-filtered pixel data.
- Pipeline Stage 10
  a. The de-blocking filter reads reconstructed data from its local memory and run de-blocking filter for H.264.
  b. The filtered pixel data, that is final reconstructed picture, is stored in local memory of de-blocking filter block for write-back.
- Pipeline Stage 11
  a. The write-back DMA writes decoded picture to external frame buffer and it is for a reference picture of next picture decoding.

Figure 69-10 shows the 10 pipeline stages for MPEG-4/H.263 encoding when the deblocking filter is disabled and Figure 69-11 shows the 11 pipeline stages for H.263 when the deblocking filter is enabled. The encoding data path in MPEG-4/H.263 is similar to the H.264 encoding data path, except for the addition of an AC/DC prediction step.

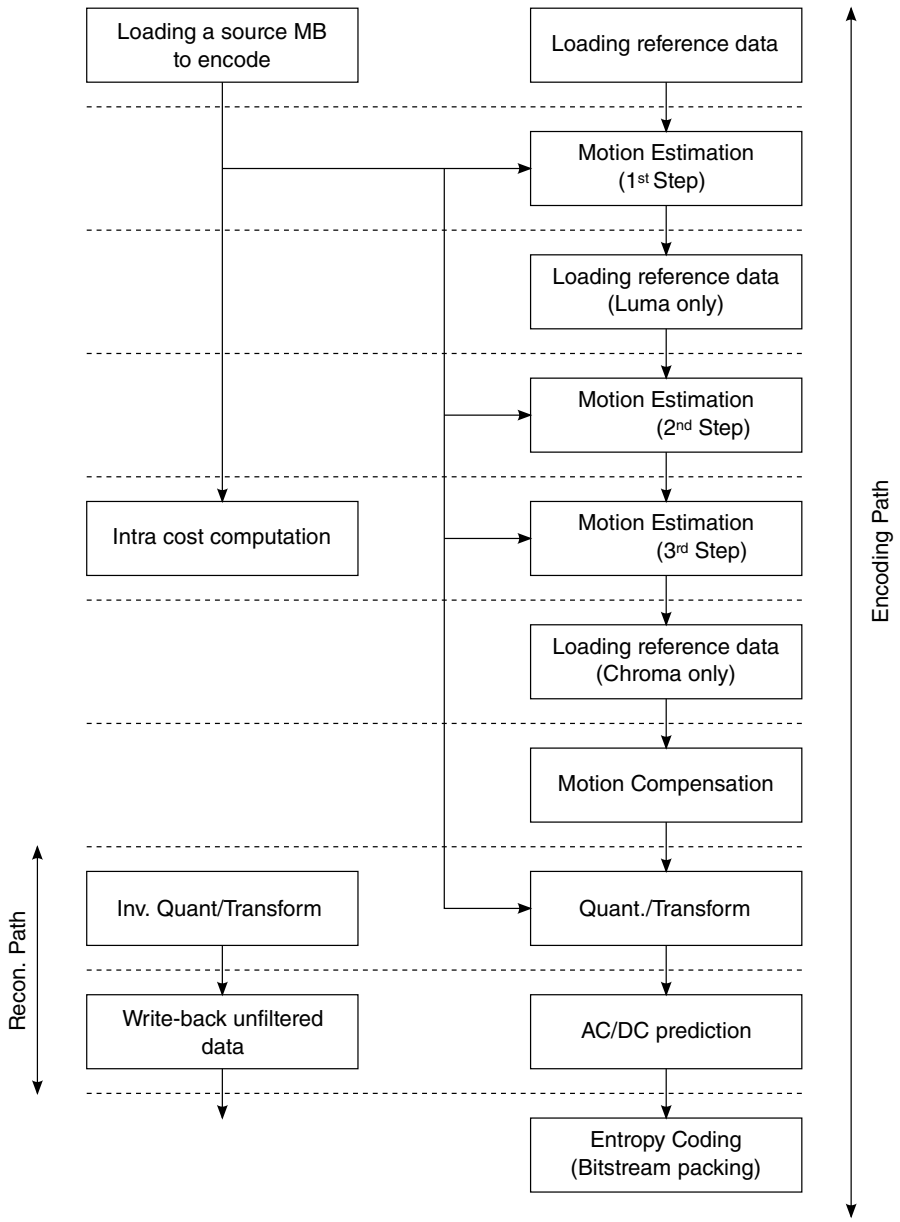


Figure 69-10. MPEG-4/H.263 encoding pipeline stage when deblocking filter is disabled

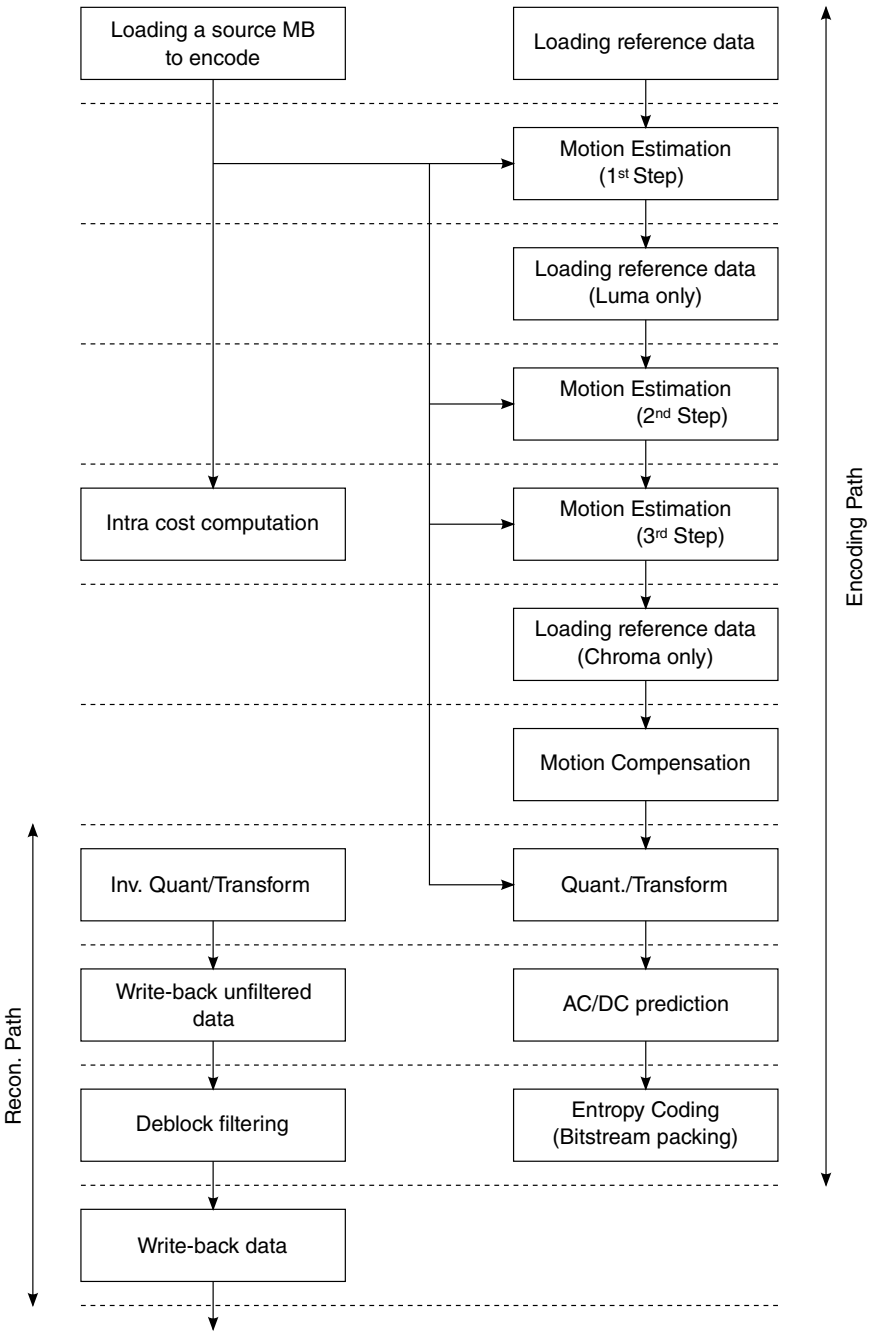


Figure 69-11. H.263 encoding pipeline stage when deblocking filter is enabled

69.7.3 Video Codec Processing Buffer Requirement

VPU has full access to the entire external memory. It uses external memory to load or store image frame, bitstream, program and data for the BIT processor.

AC/DC predication and de-blocking filtering also uses external memory. The buffer size requirement is dependent on the standard and target applications. For example, the H.264 decoding uses multiple reference frames up to 16. MPEG-4 and H.263 decoding uses only one reference frame. Each standard requires a different temporary memory size when it processes de-blocking or overlap-smoothing filtering.

VPU uses five kinds of buffers, as follows:

- Frame Buffer: for storing image frame.
- BIT processor program memory: for boot code and firmware.
- Working Buffer: for intermediate data from the BIT processor and the video decoding hardware.
- Bitstream Buffer: for loading bitstream.
- Parameter Buffer: for BIT processor command execution argument and return data.
- Search RAM: for the use of ME to reduce SDRAM bus loading.

Different buffers can be noncontiguous in external memory, though each buffer must be contiguous.

VPU also supports an optional secondary AXI bus which is connected to internal SRAM for storing temporal data of some sub-blocks, such as de-blocking filter and intra prediction. This decreases the total bandwidth to the external memory.

69.7.3.1 Memory Map Types of Frame Buffer

There are 7 map types of frame buffer:

- Type 0 : linear map
- Type 1 : Frame based tiled map, horizontal addressing
- Type 2 : Frame based tiled map, vertical addressing
- Type 3 : Field based tiled map, vertical addressing
- Type 4 : Frame/Field mixed tiled map, vertical addressing
- Type 5 : Tiled MB Raster Frame Map
- Type 6 : Tiled MB Raster Field Map

In the chip, only Type 0, 5 and 6 are supported.

In the linear map, a pixel is read out from or written to the frame buffer in a raster scan order for a frame. However, VPU requires pixels on an MB basis and such address access cannot be continuous on the physical memory. Meanwhile in the tiled map, the whole memory region is logically split into a specific number of tile-shaped segments. A tile can be an access unit in which addresses are sequential and have efficiently accessible

structure. The size of tile can be configurable according to applications. It might be an MB at least or a group of MBs whose addresses belong to the same row and bank address and are consecutive column address in SDRAM.

Meanwhile in the tiled map, the whole memory region is logically split into a specific number of tile-shaped segments. A tile can be an access unit in which addresses are sequential and have efficiently accessible structure. The size of tile can be configurable according to applications. It might be an MBat least or a group of MBs whose addresses belong to the same row and bank address and are consecutive column address in SDRAM. For more details, please refer to VPU API document.

69.7.3.2 Frame Buffer

This section describes the memory map of the frame buffer and the size requirement.

- Only 4:2:0 for all standards except (M)JPEG codec: in case of MJPEG, 4:0:0, 4:2:0, 4:2:2, and 4:4:4 are supported.
- Luminance, Cb and Cr frame buffer base addresses in 8-byte alignment. In case of the interleaved Cb/Cr map, a base address for Cr is ignored because a base address for the Cb is used to store or load the interleaved Cb/Cr samples.
- Stride for the width of the luminance frame buffer should be equal or greater than the width of picture and multiple of 8.
- Endian of the frame buffers is configurable as little or big endian.
- The internal registers for specifying a frame buffer and a picture size are 12-bit width.
- Both the interleaved Cb/Cr map and the separate Cb/Cr map are supported.
- A field pair is stored in a single frame buffer.

As shown in Figure 69-12, a frame buffer is specified with the base address and the stride line. A complete image consists of Y, U, and V components. There are two kinds of configuration, separate Cb/Cr map and interleaved Cb/Cr map.

For the separate Cb/Cr configuration, each chroma component has its own buffer. Therefore, one frame buffer needs 3 buffers for Y, U, and V components, and these buffers can be noncontiguous in memory.

For the interleaved Cb/Cr configuration, only one buffer is needed for chroma components. Therefore, one frame buffer needs 2 buffers for Y, U, and V components, and these buffers can be noncontiguous in memory.

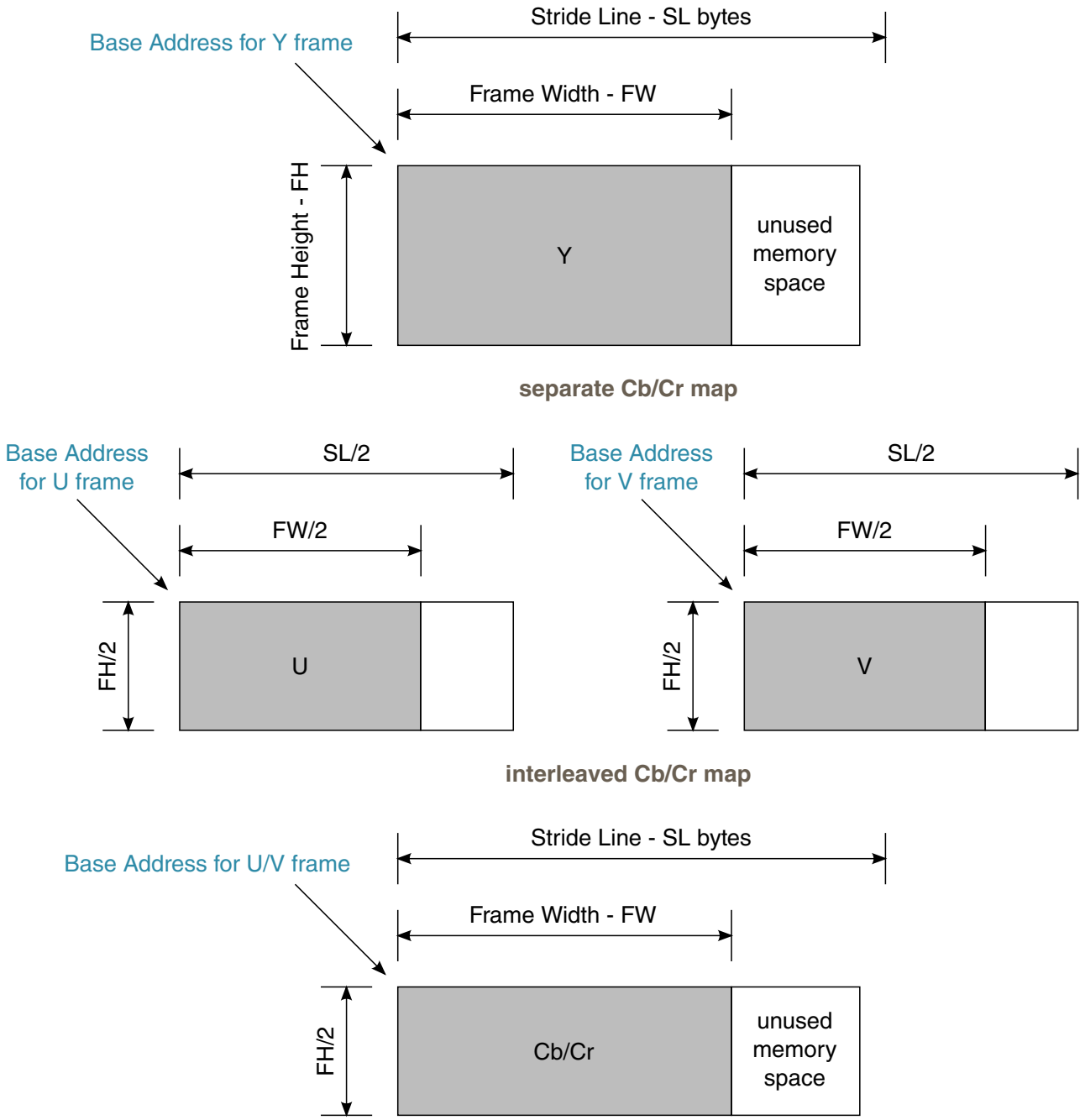


Figure 69-12. Frame Buffer configuration

Figure 69-12 shows the memory map of the frame buffer. In separate Cb/Cr map, for V frame buffer, the memory map is the same as the U frame buffer except for the base address.

VPU supports both little and big endian systems. Y(0,0) could be located in the bit[63:56]. User can specify the endianness through VPU API.Figure 69-13 gives a detailed description of these configurations considering endianess by giving examples for QCIF(176x144) images with little endian mode.

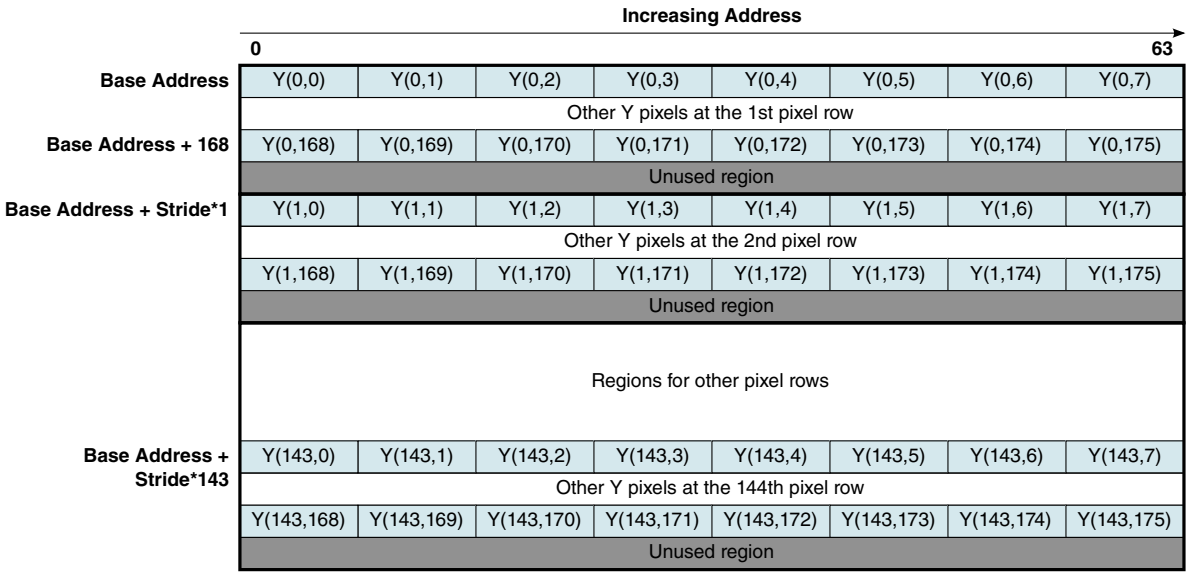


Figure 69-13. Frame Buffer Address Map of Luma in Little Endian

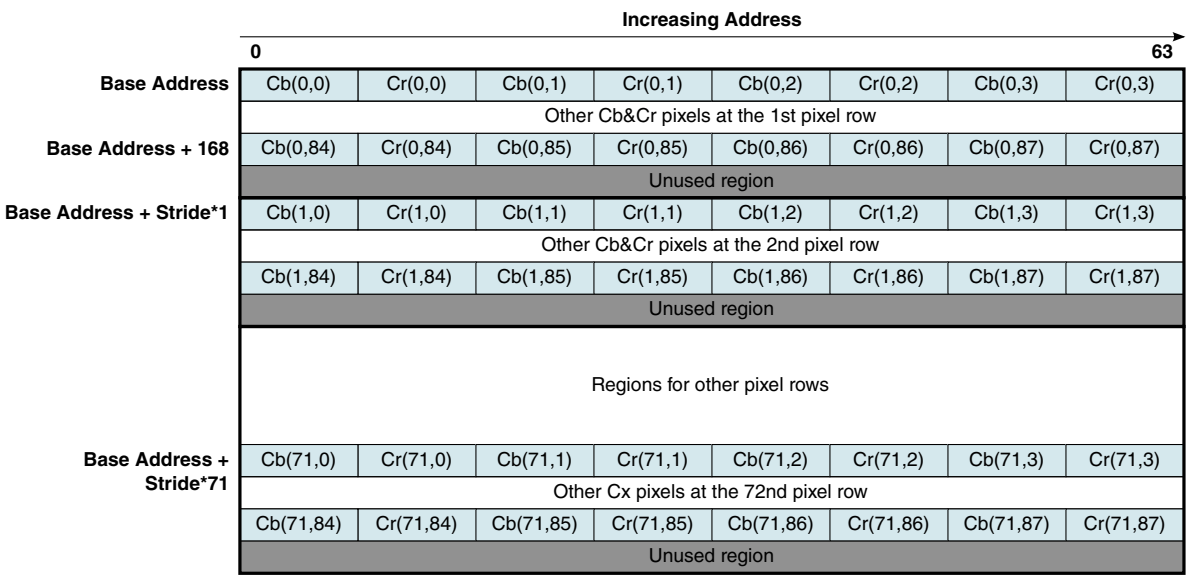


Figure 69-14. Frame Buffer Address Map of Chroma in Little Endian

The table below shows the frame buffer requirement for MPEG-4, H.264 and VC-1. The H.264 decoder requires multiple reference frames up to 16. In the case of VC-1 main profile, the decoder needs two reference frames to decode a B picture. Also, VC-1 requires two more frames that it stores for range reduction or multi-resolution.

The table below also shows the memory requirement in case of QCIF/CIF/VGA resolution image. In the case of H.264 decoding, the required size for the reference frame is dependent on the level being supported. The VPU supports up to H.264 decoding level 3.0, which the maximum decoded picture buffer size is defined as 3037.5Kbyte in the standard. To support H.264 CIF at level 3.0, 2524Kbyte is needed if 16 reference frames are used.

Table 69-4. Frame Buffer Requirement

		MEPG-4 Decoder	H.264 Decoder	VC-1 Decoder
QCIF	Reference Frames	2	16	2
	Instant Frames	1	1	2
	Display Frames	1	1	2
	Total Frames	4	18	6
	Picture Size[1]	37Kbyte	37Kbyte	37Kbyte
	Total Frame Size	148Kbyte	668Kbyte	222Kbyte
CIF	Reference Frames	2	16	2
	Instant Frames	1	1	2
	Display Frames	1	1	2
	Total Frames	4	18	6
	Picture Size	148Kbyte	148Kbyte	148Kbyte
	Total Frame Size	592Kbyte	2672Kbyte	891Kbyte
VGA	Reference Frames	2	5	2
	Instant Frames	1	1	2
	Display Frames	1	1	2
	Total Frames	4	7	6
	Picture Size	450Kbyte	450Kbyte	450Kbyte
	Total Frame Size	1800Kbyte	3150Kbyte	2700Kbyte

1. The Picture Size is the minimum size of one frame buffer with assumption that the picture is YUV 4:2:0 format and the stride line is equal to frame width.

69.7.3.3 BIT Processor Program Buffer

At the initialization stage of VPU, the host processor must download boot code to the BIT processor. After initialization, the BIT processor loads a program corresponding to the activated standard.

The program size of the boot code is 1Kbyte. There are total 112Kbyte sizes for the current version of BIT firmware to support multi-standards decoding.

69.7.3.4 Working Buffer

Besides buffers for frames and firmware, additional working buffer for intermediate data from the BIT processor and decoding is needed.

The buffers are such as the reconstructed pixel row buffer for MPEG-4 AC/DC prediction or H.264 intra-prediction, context saving buffer for running multiple processes and various temporal storage buffer for decoding process.

The required working buffer size varies according to decode size, decoding standard, decoding capability. For example, AC/DC prediction buffer size is determined by picture width and the maximum bitstream re-ordering buffer for data partition is determined by the maximum bitstream size of one picture. Working buffer size may change for different firmware version. The current version of firmware requires max 320 Kbyte for working. Its size can be set through VPU API. The detailed working buffer is organized as indicated in the table below.

Type	Name	Description	Size(KB)
STATIC	STATIC_PRC_DMEM	Bit processor data memory of each process for context switching. 5 KB for each process.	20
	STATIC_PRC_SEQ	Static data storage of each sequence. 32 KB for each process.	128
TEMP_PICMP4_DEC	MP4_DEC_ACDC	AC/DC prediction buffer of Y/Cb/Cr	6
	MP4_DEC_DP	Bitstream reordering buffer for data partition.	10
TEMP_PICAVC_DEC	AVC_DEC_IP	Intra prediction buffer of Y/Cb/Cr	72
	AVC_DEC_FMO	FMO group status buffer	6
	AVC_DEC_SLICE_INFO	Slice information buffer. Maximum 1280 slices per picture.	10
	AVC_DEC_SLICE	Slice data RBSP buffer. All slice data RBSP of one picture is stored.	116

Figure 69-15. Working buffer orgnization

69.7.3.5 Bitstream Buffer

The host processor has to assign buffers for bitstreams on a per instance basis.

If VPU handles N-bitstreams simultaneously in an application, the host should assign N bitstream buffers, and specify the base address and size. The External bitstream buffer is "ring buffer" type. The start address of ring buffer and buffer size must be written by host to BIT processor. The current read or write address of ring buffer is automatically wrapped-around by firmware.

In decoding case, the host writes the bitstream to be decoded then BIT processor reads bitstream. In this case, the bitstream overwriting or underflow may occur and if it occurs, decoding will fail. To prevent overwriting or underflow, current bitstream read/write pointer must be exchanged between the host and the BIT processor.

The BIT processor writes current read pointer of ring buffer to internal register and the host must write current write pointer of ring buffer to internal register. The BIT processor checks the bit buffer empty (underflow) status by comparing current read pointer and write pointer. If no more bitstream data is available to be decoded (buffer empty status), the BIT processor stops bitstream decoding to prevent mis-reading the bitstream and waits until the host writes more bitstream data and updates write pointer. The host must check the current read pointer and write pointer before writing more bitstream data to ring buffer to prevent overwriting bitstream data.

69.7.3.6 Parameter Buffer

Host processor must reserve parameter buffer in external memory for BIT processor command execution argument and return data.

69.7.3.7 Search RAM

VPU's motion estimation sub-block uses a search RAM to reduce the bandwidth on the external SDRAM.

Generally, motion estimation reads a reference pixel data several times. To avoid this B/W overhead, the motion estimation sub-block loads the reference pixel data form the external SDRAM one time and stores them to the search RAM through AXI bus. The stored reference pixel data is loaded several times to search the motion vector, but these operations are conducted through AMBA AXI bus. Therefore the B/W of loading reference pixel data will be reduced using search RAM in AXI bus.

69.7.3.8 Buffer Requirement Summary

Table 69-5 shows the required memory(SDRAM) size for each standard to support D1.

(720x576). These valuses are the worst case which defined by corresponding specification.

Table 69-5. Summary of Buffer Requirement

#		H.264	VC-1	AVS/RV	MPEG-4	MPEG-2
1	Frame buffer	7 frame (4242.5Kbyte)	6 frame (3.645 Kbyte)	6 frame (3.645 Kbyte)	4 frame (3430 Kbyte)	4 frame (2430 Kbyte)
2	Direct motion vector	708.75K byte	25.4Kbyte	25.4Kbyte	25.4Kbyte	
3	Overlap filter		7.1Kbyte			
4	De-blocking filter	11.25Kbyte	22.5Kbyte	11.25Kbyte		
5	Intra Prediction(AC-DC)	4.22Kbyte	5.625Kbyte	4.22Kbyte	5.625Kbyte	
6	MVP/MB information	5.625Kbyte	2.11Kbyte	2.11Kbyte	2.11Kbyte	
7	Slice information	131Kbyte				
8	Bit plane		3Kbyte			
9	Data-partioning					
	Total	5M byte	3.6M byte	3.6M byte	2.48M byte	2.38M byte

69.8 VPU Memory Map/Register Definition

VPU registers are all 32-bit wide, support only 32bit aligned read/write operation. VPU registers are grouped into several regions corresponding to different decoding process step. They are used for decoding process configuration and control. They can only be accessed through IP bus interface.

Please note that there are some undefined address space in VPU memory map, any read/write accessing to this register address space is ignored in the VPU. Read accessing to write only register returns ZERO value. Write accessing to read only register is ignored.

The BIT processor registers' memory map in VPU is 0xBASE_0000~0xBASE_01FC. The BIT processor registers are divided into 2 categories.

- Address 0xBASE_0000~0xBASE_00FC (64 registers address space) are hardware registers. These registers have reset values and their functions are fixed (not configurable).
- Address 0xBASE_0100~0xBASE_01FC (64 registers address space) are software registers. They have no reset values and can be configured by internal BIT processor. Their definitions may change for different firmware version, so they are not provided here. This type of registers can be used as general parameter registers between host processor and BIT processor.

- The first 32 parameter registers (address 0xBASE_0100~0xBASE_017C) are used as static parameters. Definition and functions of those registers are same for all kinds of run commands.
- The second 32 parameter registers (address 0xBASE_0180~0xBASE_01FC) are used as temporal parameters. The definition and functions of those registers may differ in different run commands.

The memory map for the hardware registers of VPU is shown in the table below.

Please refer to the VPU API document for descriptions on software registers access.

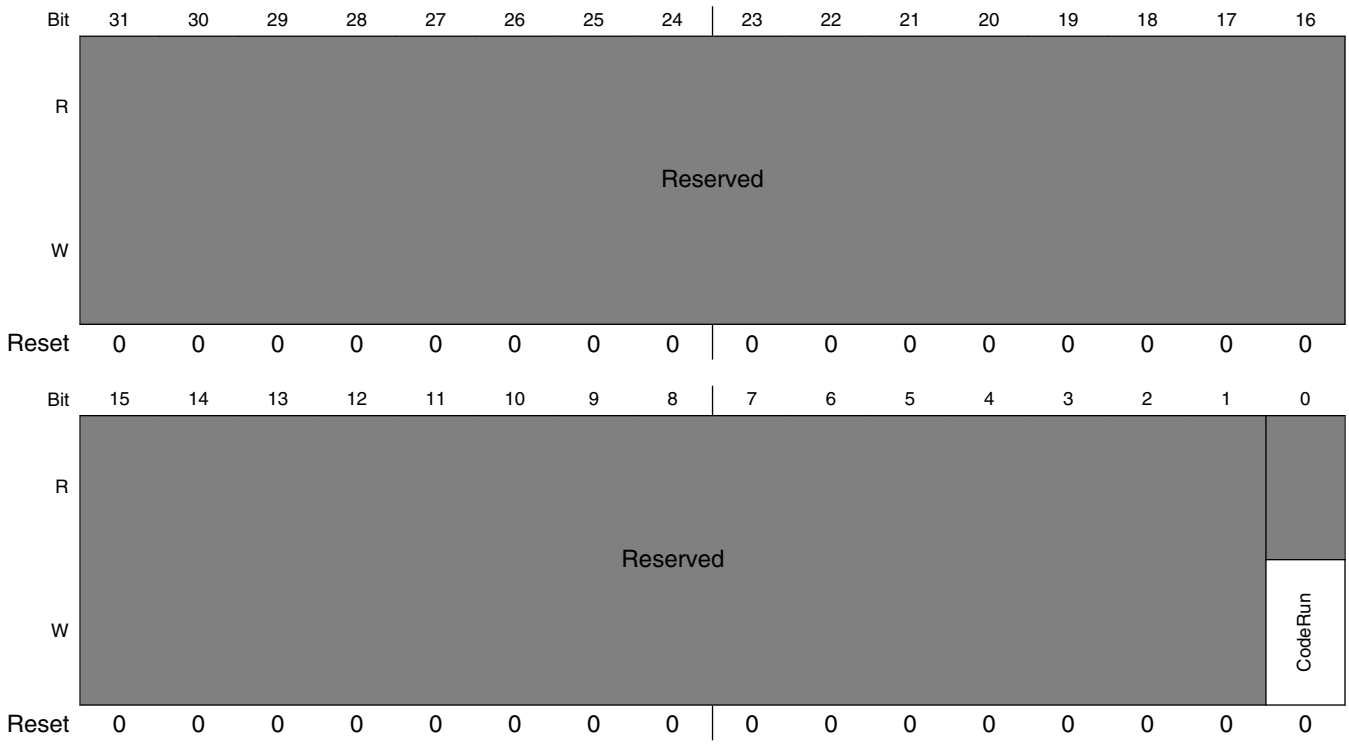
VPU memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
204_0000	BIT Processor run start (VPU_CodeRun)	32	W	0000_0000h	69.8.1/5741
204_0004	BIT Boot Code Download Data register (VPU_CodeDown)	32	W	0000_0000h	69.8.2/5741
204_0008	Host Interrupt Request to BIT (VPU_HostIntReq)	32	W	0000_0000h	69.8.3/5742
204_000C	BIT Interrupt Clear (VPU_BitIntClear)	32	W	0000_0000h	69.8.4/5743
204_0010	BIT Interrupt Status (VPU_BitIntSts)	32	R	0000_0000h	69.8.5/5744
204_0018	BIT Current PC (VPU_BitCurPc)	32	R	0000_0000h	69.8.6/5745
204_0020	BIT CODEC Busy (VPU_BitCodecBusy)	32	R	0000_0000h	69.8.7/5746

69.8.1 BIT Processor run start (VPU_CodeRun)

See the figure below for illustration of valid bits in VPU Code Run Register and the table below for description of the bit fields in the register.

Address: 204_0000h base + 0h offset = 204_0000h



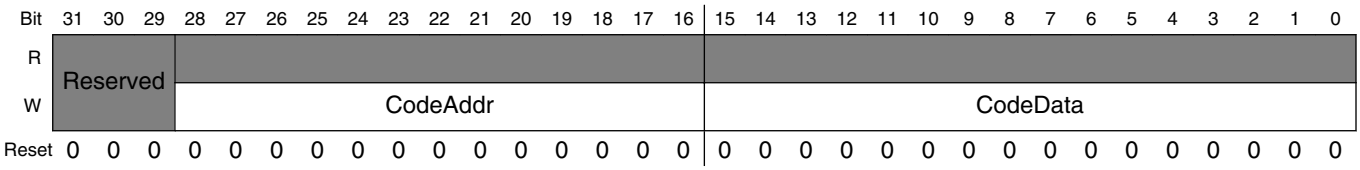
VPU_CodeRun field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 CodeRun	VPU_CodeRun. BIT processor run start bit. 0 BIT Processor stop execution. 1 BIT Processor start execution.

69.8.2 BIT Boot Code Download Data register (VPU_CodeDown)

See the figure below for illustration of valid bits in VPU BIT Boot Code Download Data Register and the following table for description of the bit fields in the register.

Address: 204_0000h base + 4h offset = 204_0004h



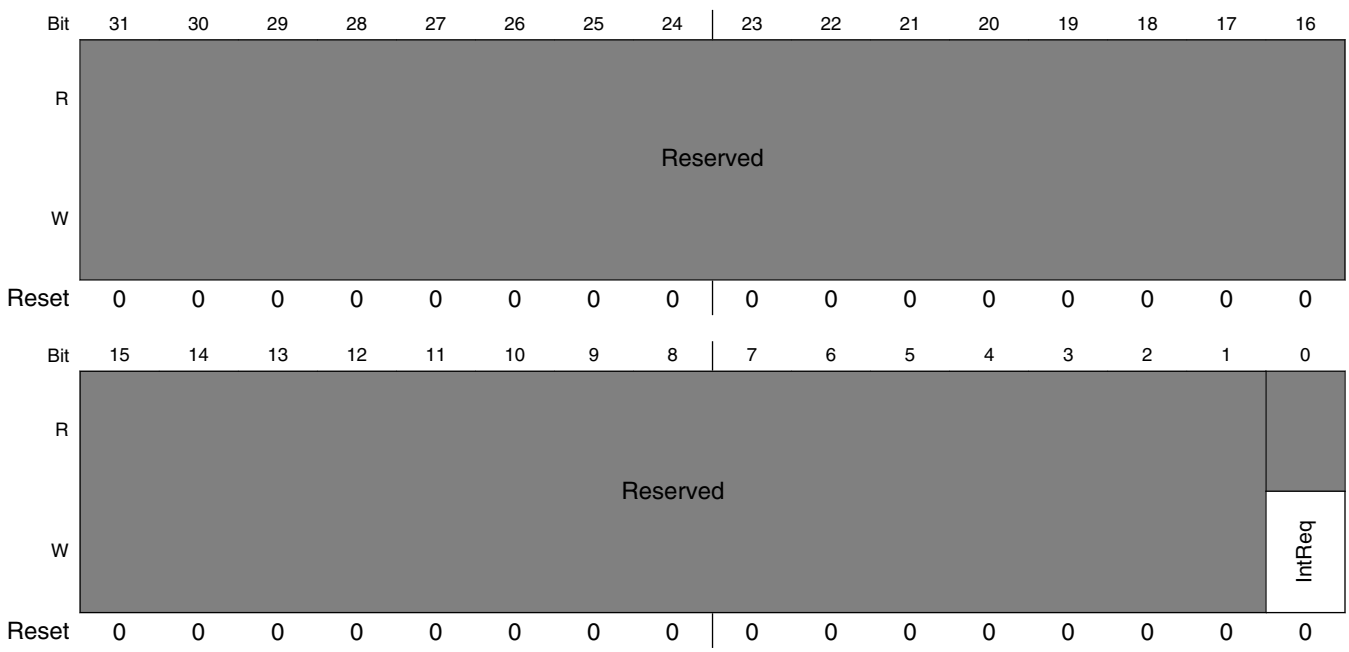
VPU_CodeDown field descriptions

Field	Description
31–29 -	This field is reserved. Reserved
28–16 CodeAddr	CodeAddr[12:0] Download address of VPU BIT boot code, which is VPU internal address of BIT processor.
CodeData	CodeData[15:0] Download data of VPU BIT boot code.

69.8.3 Host Interrupt Request to BIT (VPU_HostIntReq)

See the figure below for illustration of valid bits in VPU Host Interrupt Request Register and the following table for description of the bit fields in the register.

Address: 204_0000h base + 8h offset = 204_0008h



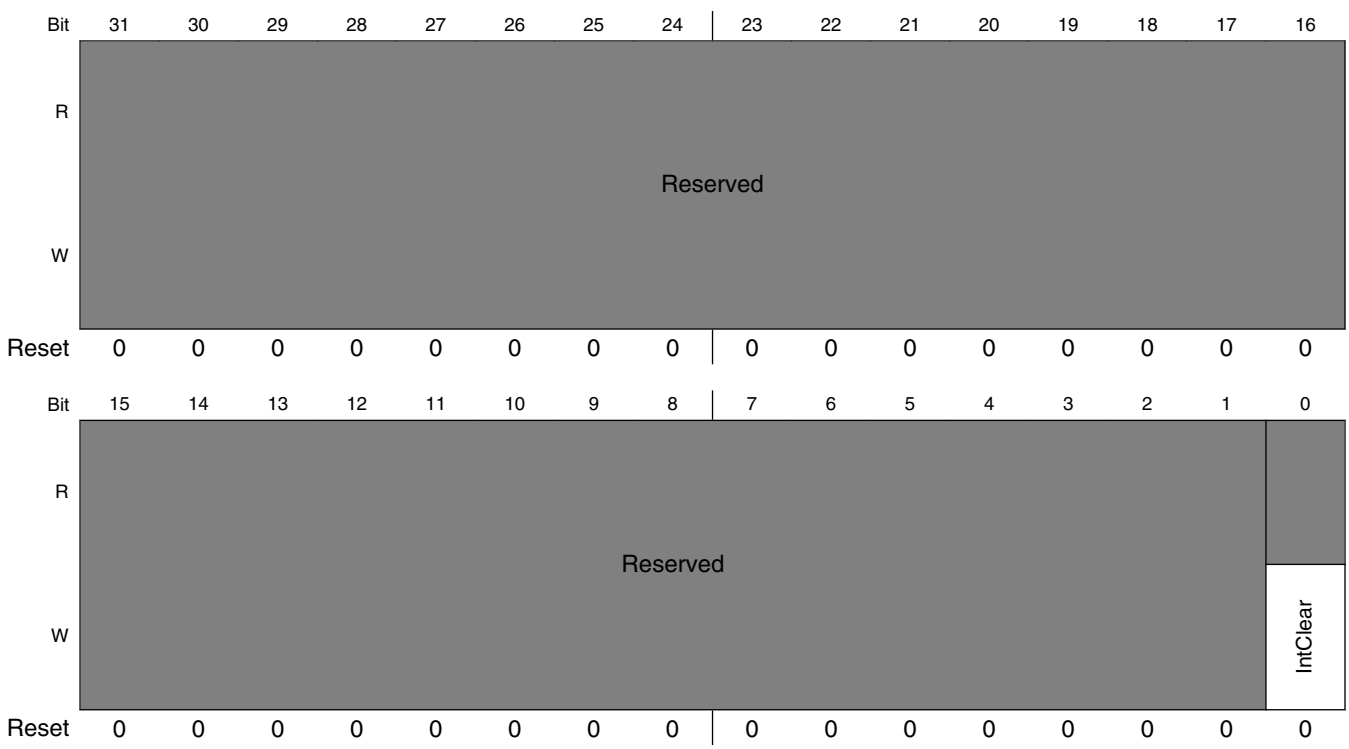
VPU_HostIntReq field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 IntReq	IntReq. The host interrupt request bit. 0 No host interrupt is requested. 1 The host processor request interrupt to the BIT processor.

69.8.4 BIT Interrupt Clear (VPU_BitIntClear)

See the figure below for illustration of valid bits in VPU BIT Interrupt Clear Register and the following table for description of the bit fields in the register.

Address: 204_0000h base + Ch offset = 204_000Ch



VPU_BitIntClear field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 IntClear	IntClear. BIT interrupt clear bit.

Table continues on the next page...

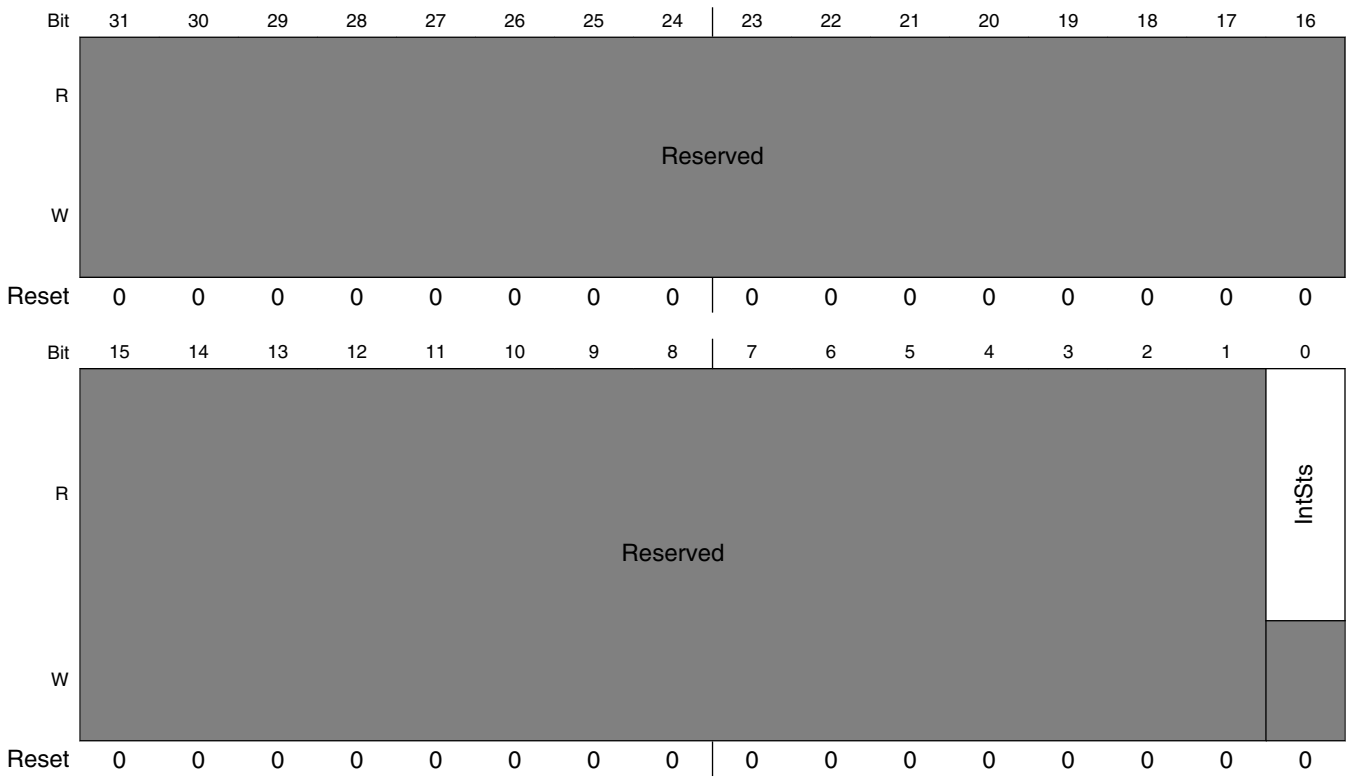
VPU_BitIntClear field descriptions (continued)

Field	Description
	0 No operation is issued. 1 Clear the BIT interrupt to the host.

69.8.5 BIT Interrupt Status (VPU_BitIntSts)

See the figure below for illustration of valid bits in VPU BIT Interrupt Status Register and the following table for description of the bit fields in the register.

Address: 204_0000h base + 10h offset = 204_0010h



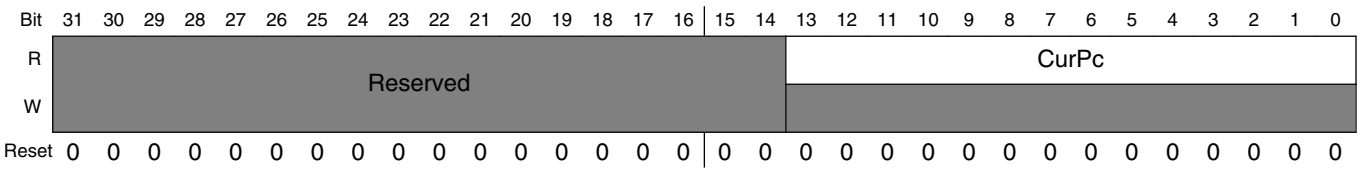
VPU_BitIntSts field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 IntSts	IntSts. BIT interrupt status bit. 0 No BIT interrupt is asserted. 1 The BIT interrupt is asserted to the host. It is cleared when the host processor write "1" to VPU_BitIntClear register.

69.8.6 BIT Current PC (VPU_BitCurPc)

See the figure below for illustration of valid bits in VPU BIT Current PC Register and the following table for description of the bit fields in the register.

Address: 204_0000h base + 18h offset = 204_0018h



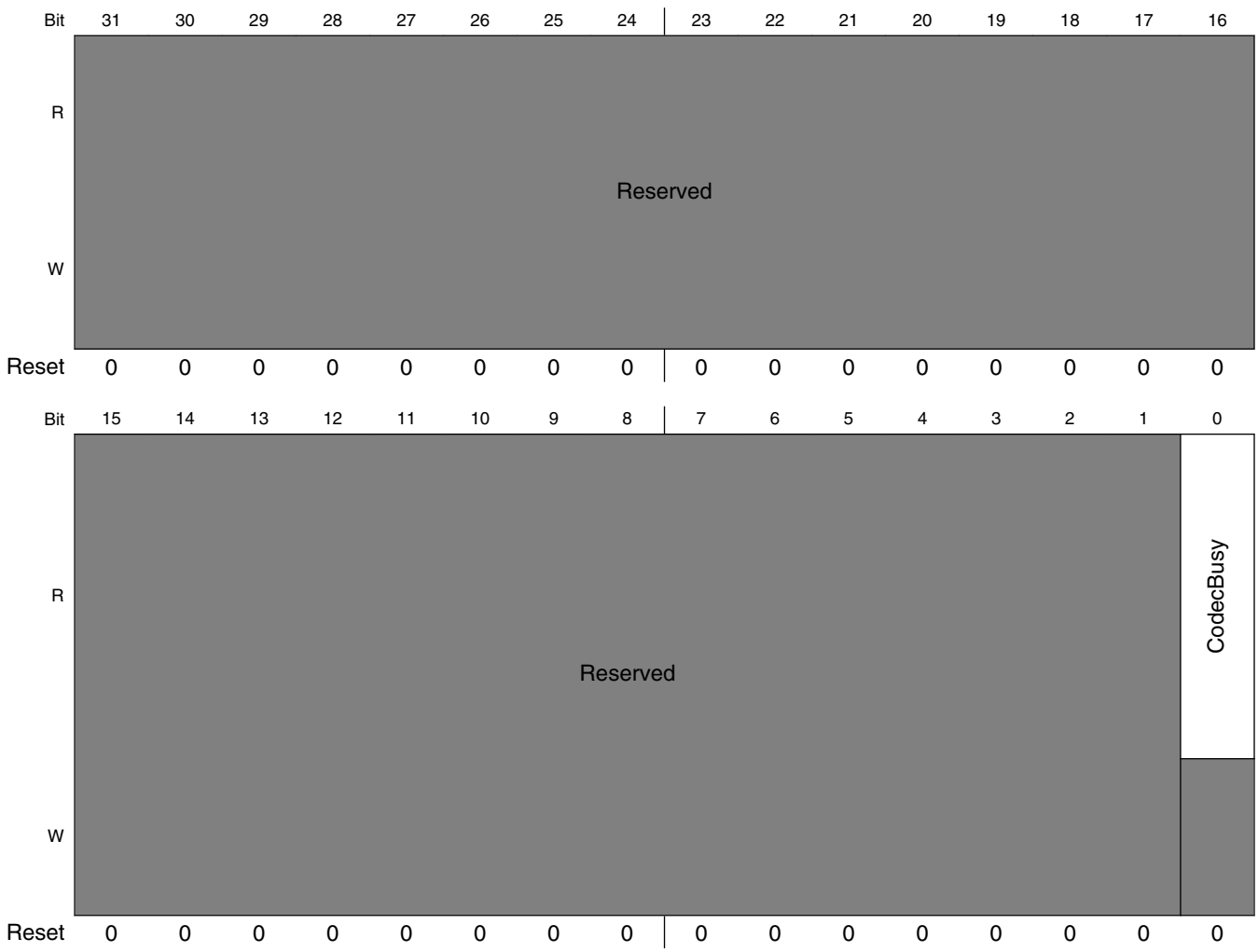
VPU_BitCurPc field descriptions

Field	Description
31–14 -	This field is reserved. Reserved
CurPc	CurPc[13:0]. BIT current PC value. Returns the current program counter of BIT processor by reading this register.

69.8.7 BIT CODEC Busy (VPU_BitCodecBusy)

See the figure below for illustration of valid bits in VPU BIT Codec Busy Register and the following table for description of the bit fields in the register.

Address: 204_0000h base + 20h offset = 204_0020h



VPU_BitCodecBusy field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 CodecBusy	Codec busy flag for Bit processor.BIT processor write "1"to this register when the processor is running."0"means processor is waiting for a command.This value is connected to the o_vpu_idle.

Chapter 70
Watchdog Timer (WDOG)

70.1 Overview

The Watchdog Timer (WDOG) protects against system failures by providing a method by which to escape from unexpected events or programming errors.

Once the WDOG is activated, it must be serviced by the software on a periodic basis. If servicing does not take place, the timer times out. Upon timeout, the WDOG asserts the internal system reset signal, WDOG_RESET_B_DEB to the System Reset Controller (SRC).

There is also a provision for WDOG signal assertion by timeout counter expiration. There is an option of programmable interrupt generation before the counter actually times out. The time at which the interrupt needs to be generated prior to counter timeout is programmable. There is a power down counter which is enabled out of any reset (POR, Warm/Cold). This counter has a fixed timeout period of 16 seconds, upon which it asserts the WDOG signal.

Flow diagrams for the timeout counter, power down counter and interrupt operations are are shown in Flow Diagrams.

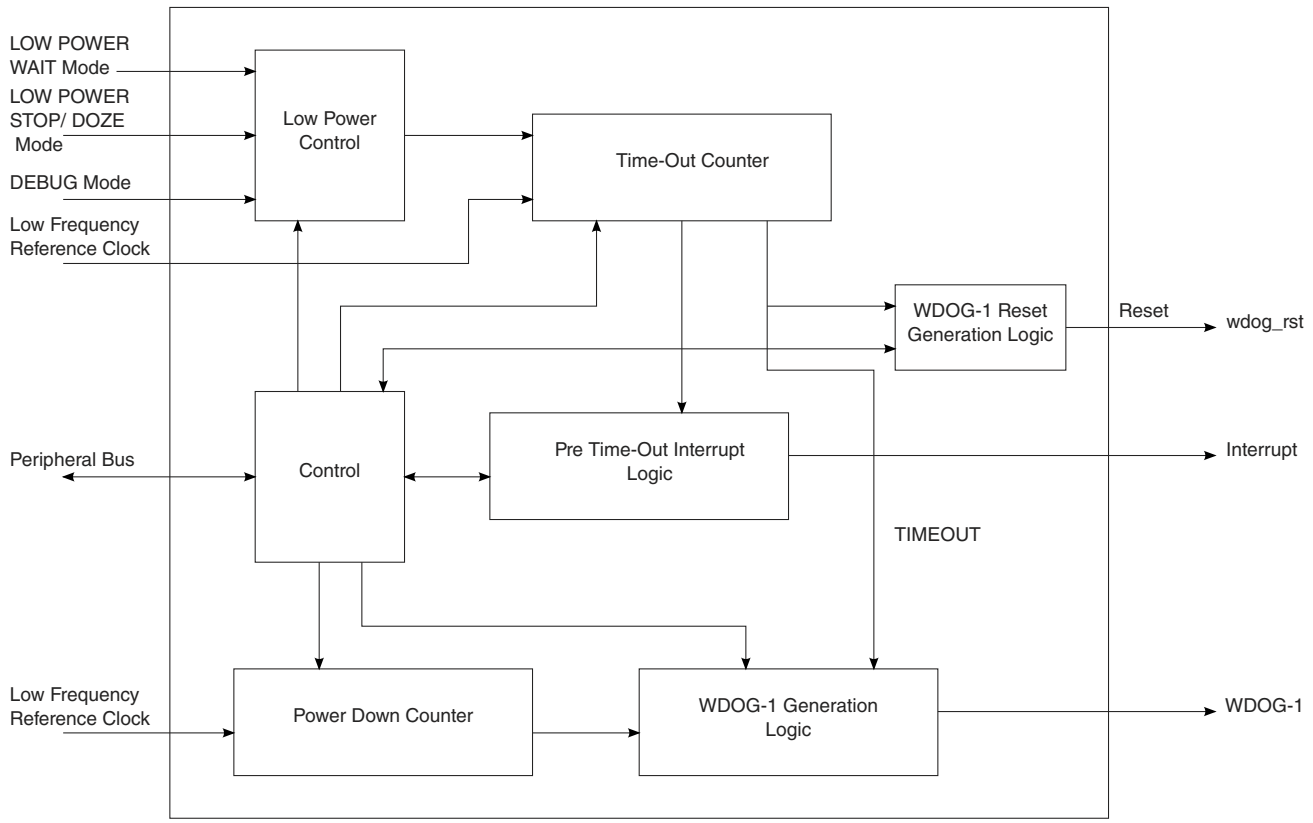


Figure 70-1. WDOG Diagram

70.1.1 Features

The WDOG features are listed below:

- Configurable timeout counter with timeout periods from 0.5 to 128 seconds which, after timeout expiration, result in the assertion of WDOG_RESET_B_DEB reset signal .
- Time resolution of 0.5 seconds
- Configurable timeout counter that can be programmed to run or stop during low-power modes
- Configurable timeout counter that can be programmed to run or stop during DEBUG mode
- Programmable interrupt generation prior to timeout
- The duration between interrupt and timeout events can be programmed from 0 to 127.5 seconds in steps of 0.5 seconds.
- Power down counter with fixed timeout period of 16 seconds, which if not disabled after reset will assert WDOG_B signal low

- Power down counter will be enabled out of any reset (POR, Warm / Cold reset) by default.

70.2 External signals

Table 70-1. WDOG External Signals

Signal	Description	Pad	Mode	Direction
WDOG1_B	This signal will power down the chip.	DISP0_DAT8	ALT3	I/O
		GPIO_9	ALT1	
		SD1_DAT2	ALT4	
WDOG1_RESET_B_DEB	This signal is a reset source for the chip.	SD1_DAT2	ALT6	O
WDOG2_B	This signal will power down the chip.	DISP0_DAT9	ALT3	I/O
		GPIO_1	ALT1	
		SD1_DAT3	ALT4	
WDOG2_RESET_B_DEB	This signal is a reset source for the chip.	SD1_DAT3	ALT6	O

70.3 Clocks

This section describes clocks and special clocking requirements of the block.

The WDOG uses the low frequency reference clock for its counter and control operations. The peripheral bus clock is used for register read/write operations.

The following table describes the clock sources for WDOG. Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 70-2. WDOG Clocks

Clock name	Clock Root	Description
ipg_clk	ipg_clk_root	IP Global functional clock. All functionality inside the WDOG module is synchronized to this clock.
ipg_clk_s	ipg_clk_root	IP slave bus clock. This clock is synchronized to ipg_clk and is only used for register read/write operations.
ipg_clk_32k	ckil_sync_clk_root	Low frequency (32.768 kHz) clock that continues to run in low-power mode. It is assumed that the Clock Controller will provide this clock signal synchronized to ipg_clk in the normal mode, and switch to a non-synchronized signal in low-power mode when the ipg_clk is off.

70.4 Watchdog mechanism and system integration

There are two WDOG modules, WDOG1 and WDOG2 (TZ) in the chip. The modules are disabled by default (after reset). WDOG1 will be configured during boot while WDOG2 is dedicated for secure world purposes and will be activated by TZ software if required. The TZ watchdog (TZ watchdog) module protects against TZ starvation by providing a method of escaping normal mode and forcing a switch to the TZ mode.TZ starvation is a situation where the normal OS prevents switching to the TZ mode. Such a situation is undesirable as it can compromise the system's security.

Once the TZ WDOG module is activated, it must be serviced by TZ on a periodic basis. If servicing does not take place, the timer times out. Upon a timeout, the TZ WDOG asserts a TZ-mapped interrupt that forces switching to the TZ mode. If it is still not serviced, the TZ WDOG asserts a security violation signal to the CSU. The TZ WDOG module cannot be programmed or de-activated by normal mode software.

The WDOG modules operate as follows:

- If servicing does not take place, the timer times out and the wdog_rst_b signal is activated (low)
- Interrupt can be generated before the counter actually times out
- The wdog_rst_b signal can be activated by software
- There is a power-down counter which gets enabled out of any reset. This counter has a fixed timeout period of 16 seconds upon which it will assert the ipp_wdog_b signal.

The following figure shows the WDOG1 and WDOG2 connectivity at the system level.

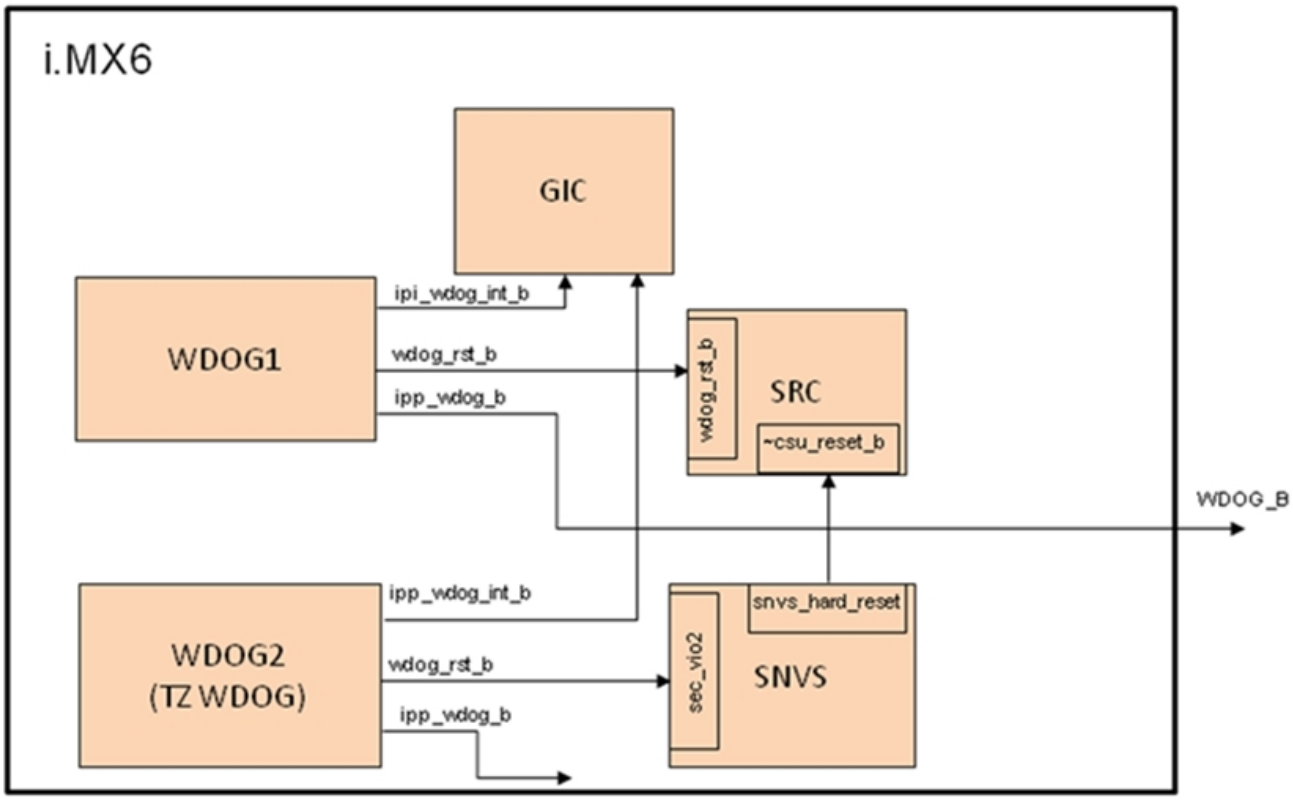


Figure 70-2. System integration

70.5 Functional description

This section provides a complete functional description of the block.

70.5.1 Timeout event

The WDOG provides timeout periods from 0.5 to 128 seconds with a time resolution of 0.5 seconds.

The user can determine the timeout period by writing to the WDOG timeout field (WT[7:0]) in the Watchdog Control Register (WDOG_WCR). The WDOG must be enabled by setting the WDE bit of Watchdog Control Register (WDOG_WCR) for the timeout counter to start running. After the WDOG is enabled, the counter is activated, loads the timeout value and begins to count down from this programmed value. The timer will time out when the counter reaches zero and the WDOG outputs a system reset signal, WDOG_RESET_B_DEB and asserts WDOG_B (WDT bit should be set in Watchdog Control Register (WDOG_WCR)).

However, the timeout condition can be prevented by reloading the counter with the new timeout value (WT[7:0] of WDOG_WCR) if a service routine (see Servicing WDOG to reload the counter) is performed before the counter reaches zero. If any system errors occur which prevent the software from servicing the Watchdog Service Register (WDOG_WSR), the timeout condition occurs. By performing the service routine, the WDOG reloads its counter to the timeout value indicated by bits WT[7:0] of the Watchdog Control Register (WDOG_WCR) and it restarts the countdown.

A system reset will reset the counter and place it in the idle state at any time during the countdown. The counter flow diagram is shown in Flow Diagrams.

NOTE

The timeout value is reloaded to the counter either at the time WDOG is enabled or after the service routine has been performed.

70.5.1.1 Servicing WDOG to reload the counter

To reload a timeout value to the counter the proper service sequence begins by writing 0x-5555 followed by 0x-AAAA to the Watchdog Service Register (WDOG_WSR). Any number of instructions can be executed between the two writes. If the WDOG_WSR is not loaded with 0x-5555 prior to writing 0x-AAAA to the WDOG_WSR, the counter is not reloaded. If any value other than 0x-AAAA is written to the WDOG_WSR after 0x-5555, the counter is not reloaded. This service sequence will reload the counter with the timeout value WT[7:0] of Watchdog Control Register (WDOG_WCR). The timeout value can be changed at any point; it is reloaded when WDOG is serviced by the core.

70.5.2 Interrupt event

Prior to timeout, the WDOG can generate an interrupt which can be considered a warning that timeout will occur shortly.

The duration between interrupt event and timeout event can be controlled by writing to the WICT field of Watchdog Interrupt Control Register (WDOG_WICR). It can vary between 0 and 127.5 seconds. If the WDOG is serviced (Servicing WDOG to reload the counter) before the interrupt generation, the counter will be reloaded with the timeout value WT[7:0] of Watchdog Control Register (WDOG_WCR) and the interrupt will not be triggered.

70.5.3 Power-down counter event

The power-down counter inside WDOG will be enabled out of reset. This counter has a fixed timeout value of 16 seconds, after which it will drive the WDOG_B signal low.

To prevent this, the software must disable this counter by clearing the PDE bit of Watchdog Miscellaneous Control Register (WDOG_WMCR) within 16 seconds of reset deassertion. Once disabled, this counter can't be enabled again until the next system reset occurs. This feature is intended to prevent the hanging up of cores after reset, as WDOG is not enabled out of reset.

70.5.4 Low power modes

70.5.4.1 STOP and DOZE mode

If the WDOG timer disable bit for low power STOP and DOZE mode (WDZST) bit in the Watchdog Control Register (WDOG_WCR), is cleared, the WDOG timer continues to operate using the low frequency reference clock. If the low power enable (WDZST) bit is set, the WDOG timer operation will be suspended in low power STOP or DOZE mode. Upon exiting low power STOP or DOZE mode, the WDOG operation returns to what it was prior to entering the STOP or DOZE mode.

70.5.4.2 WAIT mode

If the WDOG timer disable bit for low power WAIT mode (WDW) bit in the Watchdog Control Register (WDOG_WCR), is cleared, the WDOG timer continues to operate using the low frequency reference clock . If the low power WAIT enable (WDW) bit is set, the WDOG timer operation will be suspended. Upon exiting low power WAIT mode, the WDOG operation returns to what it was prior to entering the WAIT mode.

NOTE

The WDOG timer won't be able to detect events that happen for periods shorter than one low frequency reference clock cycle. For example, in repeated WAIT mode entry or exit, if the RUN mode time is less than one low frequency reference clock cycle and if the WDW bit is set, the WDOG timer may never time out, even though the system is in RUN mode for a finite duration; WDOG may not see a low frequency reference clock edge during its wake time.

70.5.5 Debug mode

The WDOG timer can be configured for continual operation, or for suspension during debug mode. If the WDOG debug enable (WDBG) bit is set in the Watchdog Control Register (WDOG_WCR), the WDOG timer operation is suspended in debug mode. If the WDBG bit is set and the debug mode is entered, WDOG timer operation is suspended after two low frequency reference clocks. Similarly, WDOG timer operation continues after two low frequency reference clocks of debug mode exit. Register read and write accesses in debug mode continue to function normally. Also, while in debug mode, the WDE bit of Watchdog Control Register (WDOG_WCR) can be enabled/disabled directly. If the WDOG debug enable (WDBG) bit is cleared then WDOG timer operation is not suspended. The power-down counter is not affected by debug mode entry/exit.

NOTE

If the WDE bit of Watchdog Control Register (WDOG_WCR) is set/cleared while in debug mode, it remains set/cleared even after exiting debug mode.

70.5.6 Operations

70.5.6.1 Watchdog reset generation

The WDOG generated reset signal WDOG_RESET_B_DEB is asserted by the following operations:

- A software write to the Software Reset Signal (SRS) bit of the Watchdog Control Register (WDOG_WCR).
- WDOG timeout. See Timeout event.

The $\overline{\text{wdog\_rst}}$ will be asserted for one clock cycle of low frequency reference clock for both a timeout condition and a software write occurrence. It remains asserted for 1 clock cycle of low frequency reference clock even if a system reset is asserted in between. Figure 70-4 shows the timing diagram of this signal due to a timeout condition.

70.5.6.2 WDOG_B generation

The WDOG asserts WDOG_B in the following scenarios:

- Software write to WDA bit of Watchdog Control Register (WDOG_WCR). WDOG_B signal remains asserted as long as the WDA bit is "0".
- WDOG timeout condition, WDT bit of Watchdog Control Register (WDOG_WCR) must be set for this scenario. A description of the timeout condition can be found in the Timeout event. WDOG_B signal remains asserted until a power-on reset (POR) occurs. It gets cleared after the POR occurs (not due to any other system reset). Figure 70-5 shows the timing diagram of WDOG_B due to timeout condition.
- WDOG power-down counter timeout, PDE bit of Watchdog Miscellaneous Control Register (WDOG_WMCR) should not be cleared for this scenario. A description of this counter can be found in the Power-down counter event. WDOG_B signal remains asserted for one clock cycle of low frequency reference clock.

Figure 70-3 shows the scenarios under which WDOG_B gets asserted.

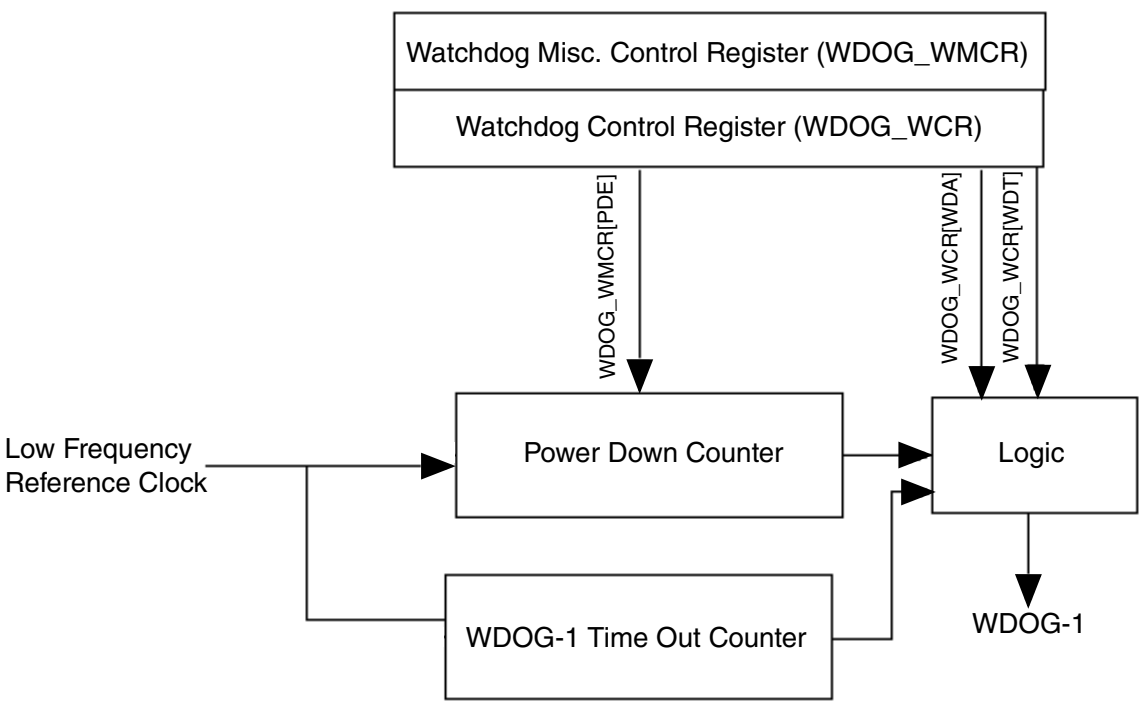


Figure 70-3. WDOG_B generation

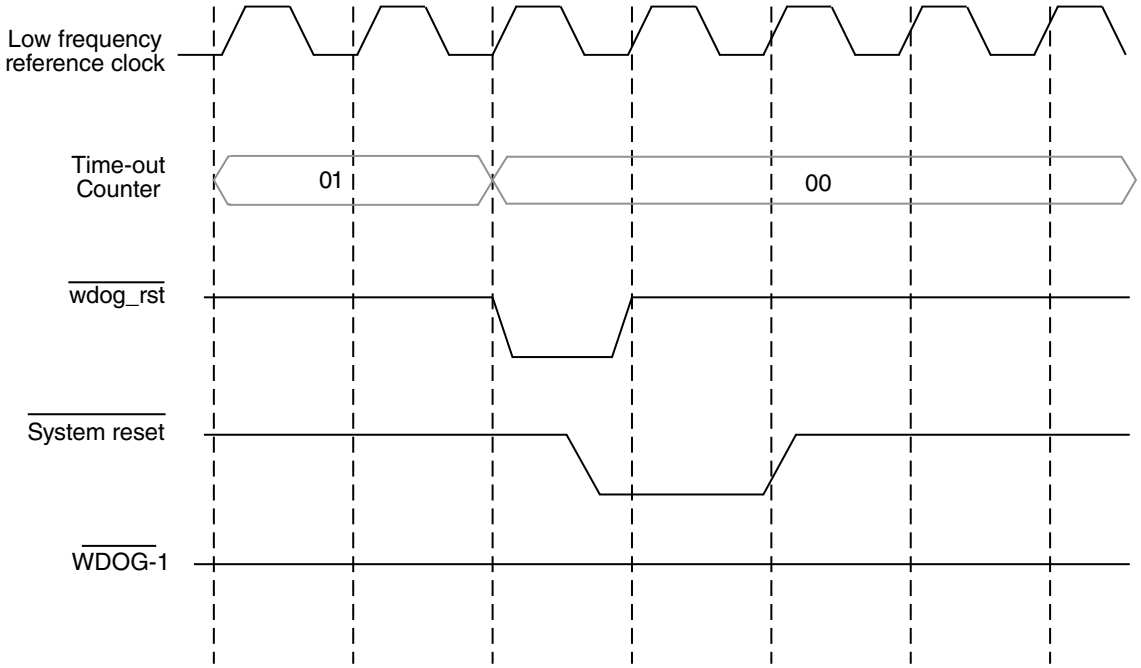


Figure 70-4. WDOG timeout vondition/WDT bit is not set

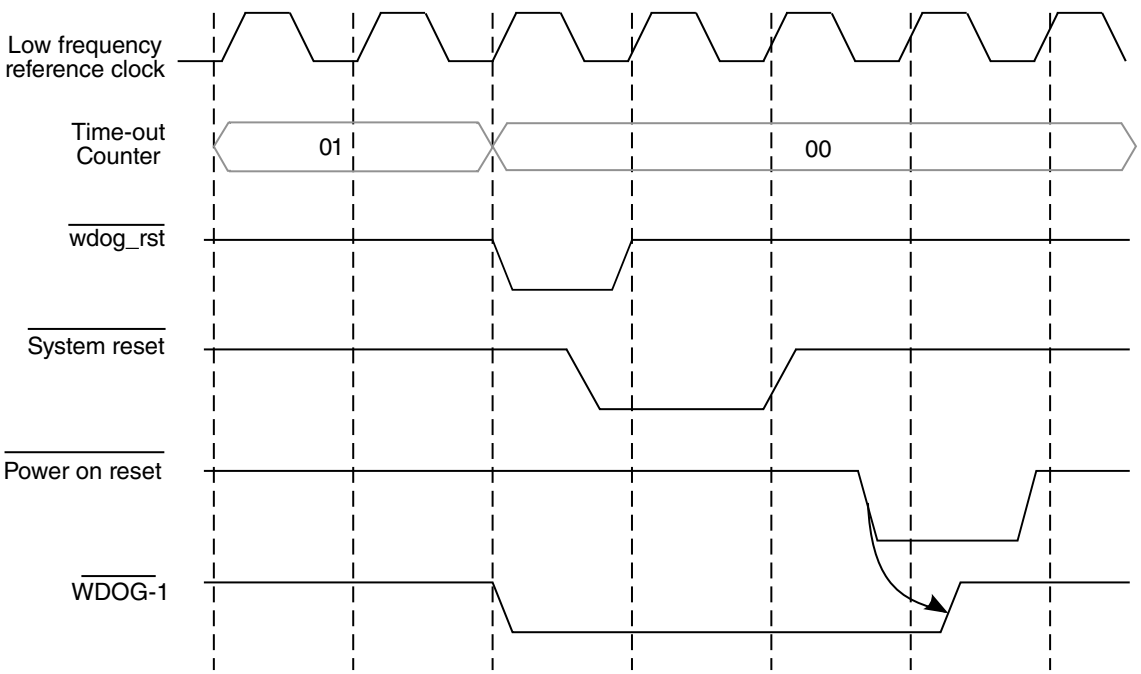


Figure 70-5. WDOG timeout condition/WDT bit is set

70.5.7 Reset

The block is reset by a system reset and the WDOG counter will be disabled. The power-down counter is enabled and starts counting.

70.5.8 Interrupt

The WDOG has the feature of Interrupt generation before timeout.

The interrupt will be generated only if the WIE bit in Watchdog Interrupt Control Register (WDOG_WICR) is set. The exact time at which the interrupt should occur (prior to timeout) depends on the value of WICT field of Watchdog Interrupt Control Register (WDOG_WICR). For example, if the WICT field has a value 0x04, then the interrupt will be generated two seconds prior to timeout. Once the interrupt is triggered the WTIS bit in Watchdog Interrupt Control Register (WDOG_WICR) will be set. The software needs to clear this bit to deassert the interrupt. If the WDOG is serviced before the interrupt generation then the counter will be reloaded with the timeout value WT[7:0] of Watchdog Control Register (WDOG_WCR) and interrupt would not be triggered.

70.5.9 Flow Diagrams

A flow diagram of WDOG operation is shown below.

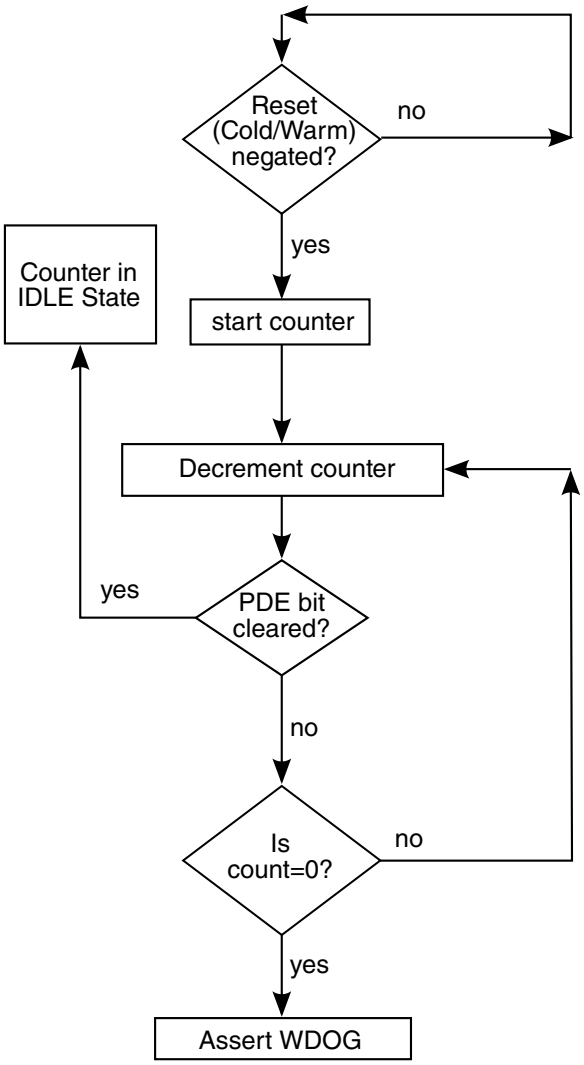


Figure 70-6. Power-Down Counter Flow Diagram

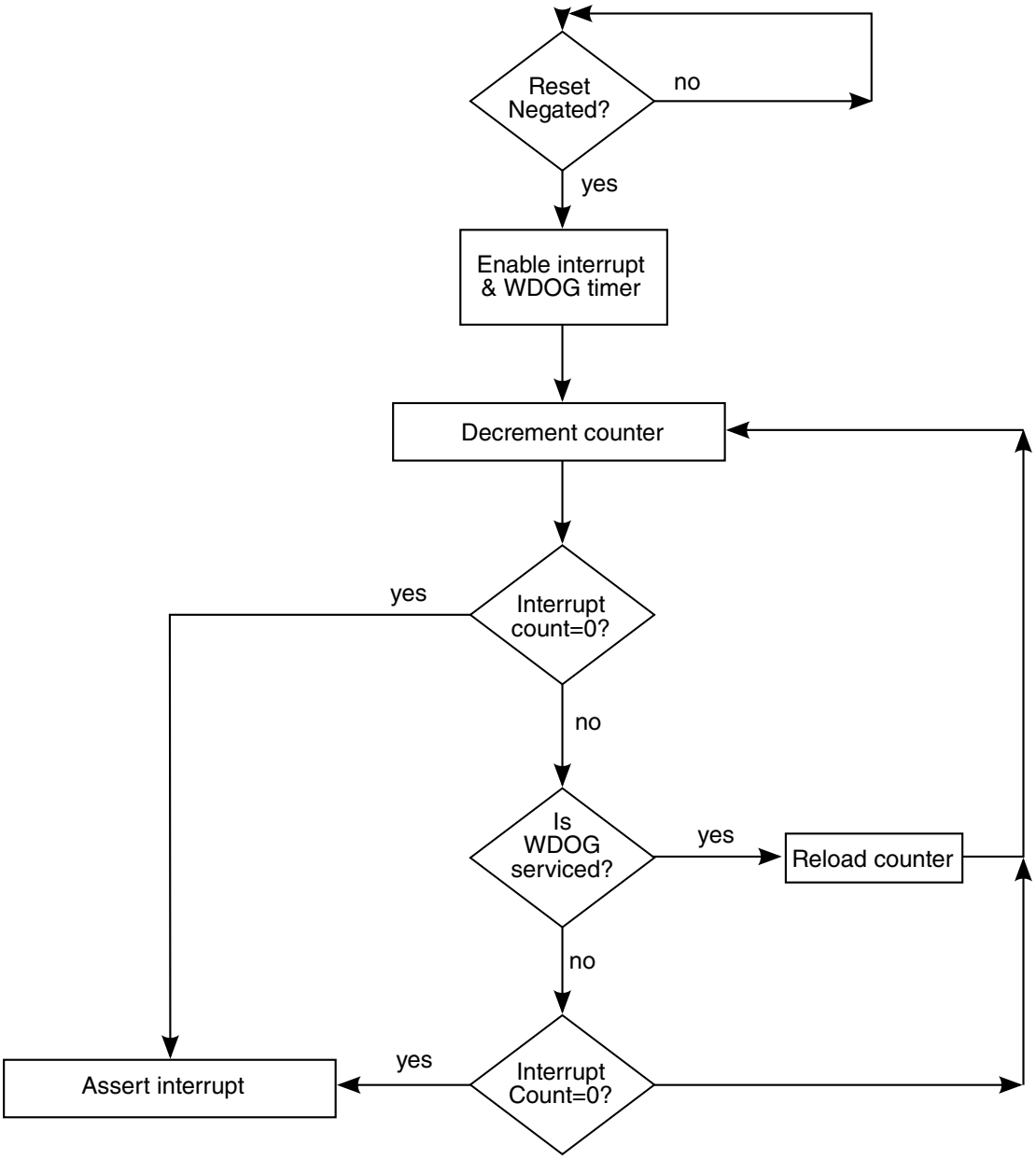


Figure 70-7. Interrupt Generation Flow Diagram

70.6 Initialization

The following sequence should be performed for WDOG initialization.

- PDE bit of Watchdog Miscellaneous Control Register (WDOG_WMCR) should be cleared to disable the power down counter.

- WDT field of Watchdog Control Register (WDOG_WCR) should be programmed for sufficient timeout value.
- WDOG should be enabled by setting WDE bit of Watchdog Control Register (WDOG_WCR) so that the timeout counter loads the WDT field value of Watchdog Control Register (WDOG_WCR) and starts counting.

70.7 WDOG Memory Map/Register Definition

The WDOG has user-accessible, 16-bit registers used to configure, operate, and monitor the state of the Watchdog Timer. Byte operations can be performed on these registers. If a 32-bit access is performed,the WDOG will not generate a peripheral bus error but will behave normally, like a 16-Bit access, making read/write possible. A 32-Bit access should be avoided, as the system may go to an unknown state.

WDOG memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20B_C000	Watchdog Control Register (WDOG1_WCR)	16	R/W	0030h	70.7.1/5760
20B_C002	Watchdog Service Register (WDOG1_WSR)	16	R/W	0000h	70.7.2/5762
20B_C004	Watchdog Reset Status Register (WDOG1_WRSR)	16	R	0000h	70.7.3/5763
20B_C006	Watchdog Interrupt Control Register (WDOG1_WICR)	16	R/W	0004h	70.7.4/5764
20B_C008	Watchdog Miscellaneous Control Register (WDOG1_WMCR)	16	R/W	0001h	70.7.5/5765
20C_0000	Watchdog Control Register (WDOG2_WCR)	16	R/W	0030h	70.7.1/5760
20C_0002	Watchdog Service Register (WDOG2_WSR)	16	R/W	0000h	70.7.2/5762
20C_0004	Watchdog Reset Status Register (WDOG2_WRSR)	16	R	0000h	70.7.3/5763
20C_0006	Watchdog Interrupt Control Register (WDOG2_WICR)	16	R/W	0004h	70.7.4/5764
20C_0008	Watchdog Miscellaneous Control Register (WDOG2_WMCR)	16	R/W	0001h	70.7.5/5765

70.7.1 Watchdog Control Register (WDOGx_WCR)

The Watchdog Control Register (WDOG_WCR) controls the WDOG operation.

- WDZST, WDBG and WDW are write-once only bits. Once the software does a write access to these bits, they will be locked and cannot be reprogrammed until the next system reset assertion.

- WDE is a write one once only bit. Once software performs a write "1" operation to this bit it cannot be reset/cleared until the next system reset.
- WDT is also a write one once only bit. Once software performs a write "1" operation to this bit it cannot be reset/cleared until the next POR. This bit does not get reset/cleared due to any system reset.

Address: Base address + 0h offset

Bit	15	14	13	12	11	10	9	8
Read Write	WT							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read Write	WDW	Reserved	WDA	SRS	WDT	WDE	WDBG	WDZST
Reset	0	0	1	1	0	0	0	0

WDOGx_WCR field descriptions

Field	Description
15–8 WT	Watchdog Time-out Field. This 8-bit field contains the time-out value that is loaded into the Watchdog counter after the service routine has been performed or after the Watchdog is enabled. After reset, WT[7:0] must have a value written to it before enabling the Watchdog otherwise count value of zero which is 0.5 seconds is loaded into the counter. **NOTE:** The time-out value can be written at any point of time but it is loaded to the counter at the time when WDOG is enabled or after the service routine has been performed. For more information see Timeout event . 0x00 - 0.5 Seconds (Default). 0x01 - 1.0 Seconds. 0x02 - 1.5 Seconds. 0x03 - 2.0 Seconds. 0xff - 128 Seconds.
7 WDW	Watchdog Disable for Wait. This bit determines the operation of WDOG during Low Power WAIT mode. This is a write once only bit. 0 Continue WDOG timer operation (Default). 1 Suspend WDOG timer operation.
6 -	Reserved This field is reserved. adopt Reserved
5 WDA	WDOG_B assertion. Controls the software assertion of the WDOG_B signal. 0 Assert WDOG_B output. 1 No effect on system (Default).
4 SRS	Software Reset Signal. Controls the software assertion of the WDOG-generated reset signal WDOG_RESET_B_DEB . This bit automatically resets to "1" after it has been asserted to "0". **NOTE:** This bit does not generate the software reset to the block. 0 Assert system reset signal. 1 No effect on the system (Default).

Table continues on the next page...

WDOGx_WCR field descriptions (continued)

Field	Description
3 WDT	WDOG_B Time-out assertion. Determines if the WDOG_B gets asserted upon a Watchdog Time-out Event. This is a write-one once only bit. **NOTE:** There is no effect on WDOG_RESET_B_DEB (WDOG Reset) upon writing on this bit. WDOG_B gets asserted along with WDOG_RESET_B_DEB if this bit is set. 0 No effect on WDOG_B (Default). 1 Assert WDOG_B upon a Watchdog Time-out event.
2 WDE	Watchdog Enable. Enables or disables the WDOG block. This is a write one once only bit. It is not possible to clear this bit by a software write, once the bit is set. **NOTE:** This bit can be set/reset in debug mode (exception). 0 Disable the Watchdog (Default). 1 Enable the Watchdog.
1 WDBG	Watchdog DEBUG Enable. Determines the operation of the WDOG during DEBUG mode. This bit is write once only. 0 Continue WDOG timer operation (Default). 1 Suspend the watchdog timer.
0 WDZST	Watchdog Low Power. Determines the operation of the WDOG during low-power modes. This bit is write once-only. **NOTE:** The WDOG can continue/suspend the timer operation in the low-power modes (STOP and DOZE mode). 0 Continue timer operation (Default). 1 Suspend the watchdog timer.

70.7.2 Watchdog Service Register (WDOGx_WSR)

When enabled, the WDOG requires that a service sequence be written to the Watchdog Service Register (WSR) to prevent the timeout condition.

NOTE

Executing the service sequence will reload the WDOG timeout counter.

Address: Base address + 2h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	WSR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

WDOGx_WSR field descriptions

Field	Description
WSR	Watchdog Service Register. This 16-bit field contains the Watchdog service sequence. Both writes must occur in the order listed prior to the time-out, but any number of instructions can be executed between the two writes. The service sequence must be performed as follows: 0x5555 Write to the Watchdog Service Register (WDOG_WSR). 0xAAAA Write to the Watchdog Service Register (WDOG_WSR).

70.7.3 Watchdog Reset Status Register (WDOGx_WRSR)

The WRSR is a read-only register that records the source of the output reset assertion. It is not cleared by a hard reset. Therefore, only one bit in the WRSR will always be asserted high. The register will always indicate the source of the last reset generated due to WDOG. Read access to this register is with one wait state. Any write performed on this register will generate a Peripheral Bus Error .

A reset can be generated by the following sources, as listed in priority from highest to lowest:

- Watchdog Time-out
- Software Reset

Address: Base address + 4h offset

Bit	15	14	13	12	11	10	9	8
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0			POR	0		TOUT	SFTW
Write								
Reset	0	0	0	0	0	0	0	0

WDOGx_WRSR field descriptions

Field	Description
15–5 Reserved	This read-only field is reserved and always has the value 0.
4 POR	Power On Reset. Indicates whether the reset is the result of a power on reset. 0 Reset is not the result of a power on reset. 1 Reset is the result of a power on reset.
3–2 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

WDOGx_WRSR field descriptions (continued)

Field	Description
1 TOUT	Timeout. Indicates whether the reset is the result of a WDOG timeout. 0 Reset is not the result of a WDOG timeout. 1 Reset is the result of a WDOG timeout.
0 SFTW	Software Reset. Indicates whether the reset is the result of a WDOG software reset by asserting SRS bit 0 Reset is not the result of a software reset. 1 Reset is the result of a software reset.

70.7.4 Watchdog Interrupt Control Register (WDOGx_WICR)

The WDOG_WICR controls the WDOG interrupt generation.

Address: Base address + 6h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	WIE	WTIS	0						WICT							
Write		w1c														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

WDOGx_WICR field descriptions

Field	Description
15 WIE	Watchdog Timer Interrupt enable bit. Reset value is 0. **NOTE:** This bit is a write once only bit. Once the software does a write access to this bit, it will get locked and cannot be reprogrammed until the next system reset assertion 0 Disable Interrupt (Default). 1 Enable Interrupt.
14 WTIS	Watchdog TImer Interrupt Status bit will reflect the timer interrupt status, whether interrupt has occurred or not. Once the interrupt has been triggered software must clear this bit by writing 1 to it. 0 No interrupt has occurred (Default). 1 Interrupt has occurred
13–8 Reserved	This read-only field is reserved and always has the value 0.
WICT	Watchdog Interrupt Count Time-out (WICT) field determines, how long before the counter time-out must the interrupt occur. The reset value is 0x04 implies interrupt will occur 2 seconds before time-out. The maximum value that can be programmed to WICT field is 127.5 seconds with a resolution of 0.5 seconds. **NOTE:** This field is write once only. Once the software does a write access to this field, it will get locked and cannot be reprogrammed until the next system reset assertion. 0x00 WICT[7:0] = Time duration between interrupt and time-out is 0 seconds. 0x01 WICT[7:0] = Time duration between interrupt and time-out is 0.5 seconds. 0x04 WICT[7:0] = Time duration between interrupt and time-out is 2 seconds (Default). 0xff WICT[7:0] = Time duration between interrupt and time-out is 127.5 seconds.

70.7.5 Watchdog Miscellaneous Control Register (WDOGx_WMCR)

WDOG_WMCR Controls the Power Down counter operation.

Address: Base address + 8h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0															PDE
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

WDOGx_WMCR field descriptions

Field	Description
15–1 Reserved	This read-only field is reserved and always has the value 0.
0 PDE	Power Down Enable bit. Reset value of this bit is 1, which means the power down counter inside the WDOG is enabled after reset. The software must write 0 to this bit to disable the counter within 16 seconds of reset de-assertion. Once disabled this counter cannot be enabled again. See Power-down counter event for operation of this counter. **NOTE:** This bit is write-one once only bit. Once software sets this bit it cannot be reset until the next system reset. 0 Power Down Counter of WDOG is disabled. 1 Power Down Counter of WDOG is enabled (Default).

Chapter 71
Crystal Oscillator (XTALOSC)

71.1 Overview

This block comprises both the 24 MHz and 32 kHz implementation of a biased amplifier that when combined with a suitable external quartz crystal and external load capacitors, implements an oscillator.

The block includes means to:

- Accept an external clock source.
- Detect if the crystal frequency is close to 24 MHz or 32 kHz.
- Reduce the operating current via software after the oscillator has started (24 MHz specific feature)
- Supply another ~32 kHz clock source based off an independent internal oscillator if there is no oscillation sensed on the RTC_XTAL bumps(contacts) (32 kHz specific feature). The internal oscillator will provide clocks to the same on-chip modules as the external 32 kHz oscillator.
- Automatically switch to the external oscillation source when sensed on the RTC_XTAL bumps(contacts) (32 kHz specific feature).

71.2 External Signals

Thetable found here describes the external signals of XTALOSC:

Table 71-1. XTALOSC External Signals

Signal	Description	Pad	Mode	Direction
CLK1_N	Negative differential clock signal	CLK1_N	No Muxing	O
CLK1_P	Positive differential clock signal	CLK1_P	No Muxing	O
CLK2_N	Negative differential clock signal 2	CLK2_N	No Muxing	O
CLK2_P	Positive differential clock signal 2	CLK2_P	No Muxing	O

Table continues on the next page...

Table 71-1. XTALOSC External Signals (continued)

Signal	Description	Pad	Mode	Direction
REF_CLK_24M	24 MHz reference clock	GPIO_3	ALT3	O
		RGMII_TXC	ALT7	
REF_CLK_32K	32 KHz reference clock	GPIO_8	ALT1	O
		NVCC_SD2	No Muxing	
RTC_XTALI	Real-time clock crystal oscillator input	RTC_XTALI	No Muxing	I
RTC_XTALO	Real-time clock crystal oscillator output	RTC_XTALO	No Muxing	O
XTALI	Crystal oscillator input signal	XTALI	No Muxing	I
XTALO	Crystal oscillator output signal	XTALO	No Muxing	O

71.3 Crystal Oscillator 24 MHz

71.3.1 Oscillator Configuration (24 MHz)

The basic block diagram of the 24 MHz module configured as a crystal oscillator is shown below.

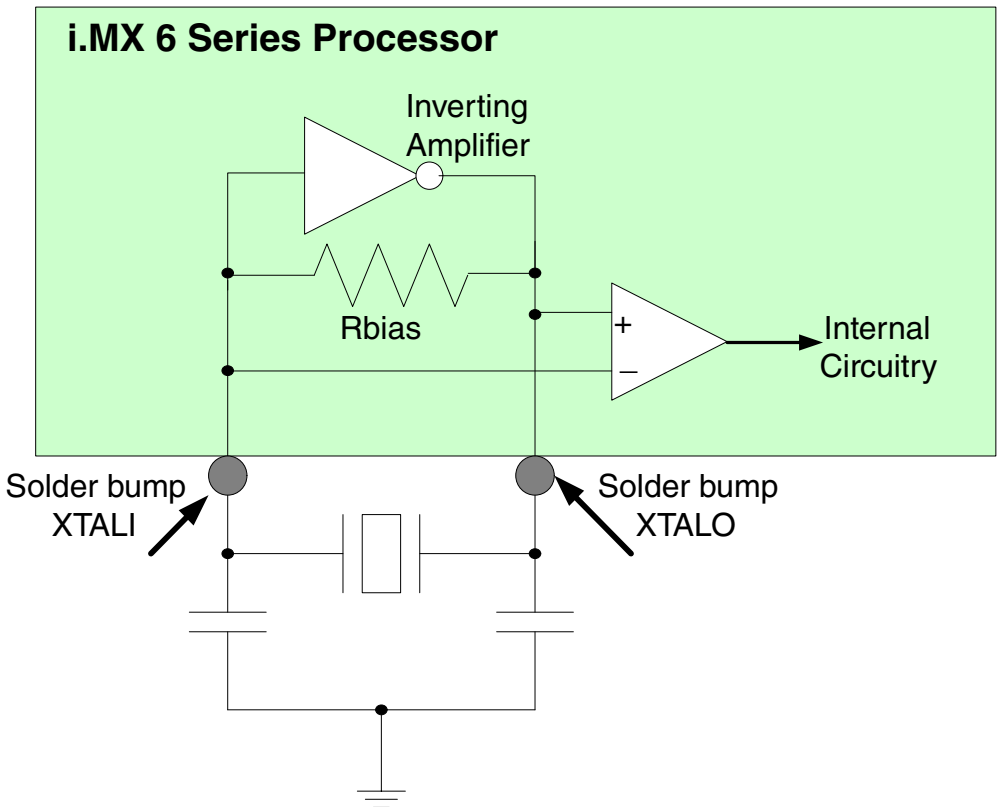


Figure 71-1. Oscillator Configuration (24 MHz)

This integrated biased amplifier can be used to create different frequency oscillators with different external component selection. However, care should be taken as many of the serial IO modules depend on the fixed frequency of 24 MHz. Please consult the sections of the document pertaining to the USB, ENET, PCIe, and SATA interfaces, for example. Once a healthy oscillation is established, then the bias current of the oscillator can generally be reduced to save power. This is accomplished through the XTALOSC24M_MISC0[OSC_I] bits, defined in the MISC0 register later in this chapter. Restore the XTALOSC24M_MISC0[OSC_I] bits before going into a power mode where the XTALOSC24 is powered down or oscillator startup may become an issue. The power down of the XTALOSC24 module is controlled by the CCM. See this section of the manual for more details.

71.3.2 Bypass Configuration (24 MHz)

If it is desired to drive the chip with an external clock source, then the 24 MHz oscillator could be driven in one of three configurations using a nominal 1.1V source.

1. A single ended external clock source can be used to overdrive the output of the amplifier (XTALO). Since the oscillation sensing amplifier is differential, the

XTALI pin should be externally floating and capacitively loaded. The combination of the internal biasing resistor and the external capacitor will filter the signal applied to the XTALO pin and develop a rough reference for the sensing amplifier to compare to.

2. A single ended external clock source can be used to drive XTALI. In this configuration, XTALO should be left externally floating.
3. A differential external clock source can be used to drive both XTALI and XTALO.

Generally, configuration 2 is anticipated to be the most used configuration, but all three configurations may be utilized.

71.3.3 Crystal Frequency Detection(24 MHz)

A submodule exists that gives a fairly crude (relative to the accuracy of a crystal) estimation of whether the clock frequency is correct.

This function may be enabled by setting the XTALOSC24M_MISC0[OSC_XTALOK_EN] bit. It is disabled at system reset. When the oscillator is stable and the correct frequency is detected, the XTALOSC24M_MISC0[OSC_XTALOK] bit will be set. Note that the correct frequency will be observed before the oscillator fully blooms(the oscillation waveform build-up is completed).

71.4 Crystal Oscillator 32 kHz

71.4.1 Oscillator Configuration (32 kHz)

The basic block diagram of the 32 kHz module configured as a crystal oscillator is shown below.

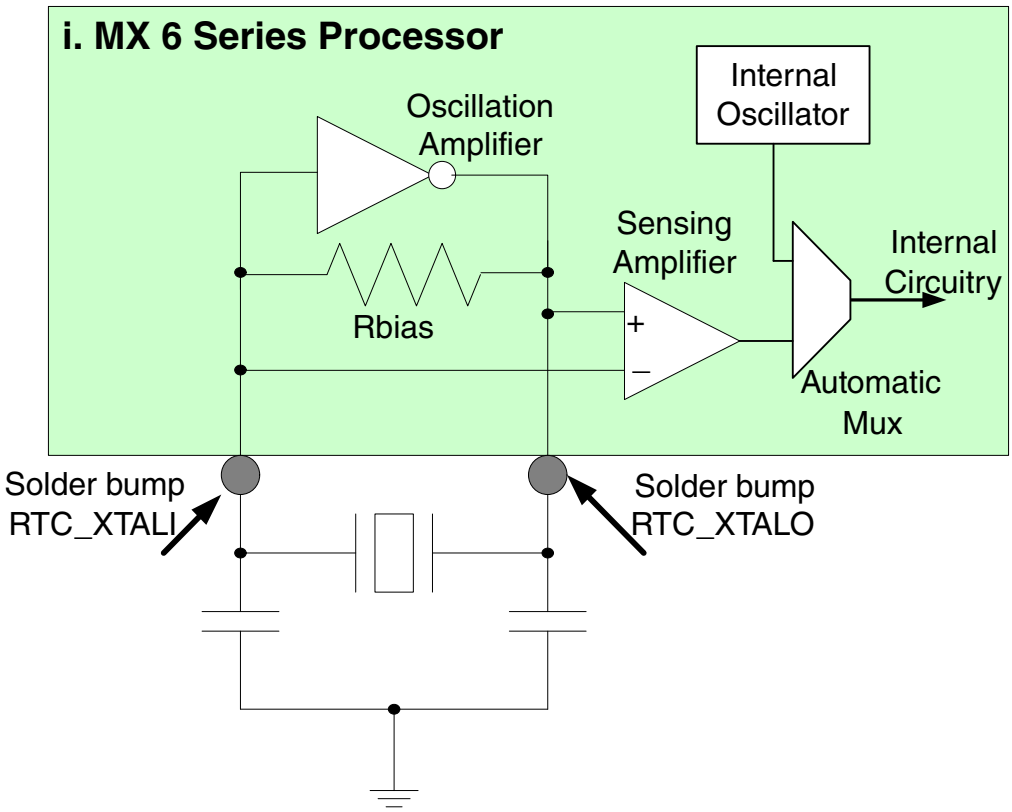


Figure 71-2. Oscillator Configuration (32 kHz)

This integrated biased amplifier can be used to create different frequency oscillators with different external component selection. Generally, RTC oscillators are either implemented with 32 kHz or 32.768 kHz crystals. Please consult the Security Reference Manual for appropriate frequency selection and configuration. Care must be taken to limit external leakage as this may debias the amplifier and degrade the gain.

The internal oscillator is automatically multiplexed in the clocking system when the system detects a loss of clock. The internal oscillator will provide clocks to the same on-chip modules as the external 32 kHz oscillator. The internal oscillator is not precise relative to a crystal. While it will provide a clock to the system, it generally will not be precise enough for long term time keeping. The internal oscillator is anticipated to be useful for quicker startup times, tampering prevention, and for cost savings in applications not needing the precision of an external crystal.

71.4.2 Bypass Configuration (32 kHz)

If it is desired to drive the chip with an external clock source, then the 32 kHz oscillator could be driven in one of three configurations using a nominal 1.1V source.

1. A single ended external clock source can be used to overdrive the output of the amplifier (RTC_XTALO). Since the oscillation sensing amplifier is differential, the RTC_XTALI pin should be externally floating and capacitively loaded. The combination of the internal biasing resistor and the external capacitor will filter the signal applied to the RTC_XTALO pin and develop a rough reference for the sensing amplifier to compare to.
2. A single ended external clock source can be used to drive RTC_XTALI. In this configuration, RTC_XTALO should be left externally floating.
3. A differential external clock source can be used to drive both RTC_XTALI and RTC_XTALO.

Generally, configuration 2 is anticipated to be the most used configuration, but all three configurations may be utilized.

71.5 XTALOSC 24MHz Memory Map/Register Definition

NOTE

The register content is mixed with analog functions not related to the oscillator function. These bits are noted.

XTALOSC24M memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20C_8150	Miscellaneous Register 0 (XTALOSC24M_MISC0)	32	R/W	0400_0000h	71.5.1/5773

71.5.1 Miscellaneous Register 0 (XTALOSC24M_MISC0)

This register defines the control and status bits for miscellaneous analog blocks.

Address: 20C_8000h base + 150h offset = 20C_8150h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved			CLKGATE_DELAY			CLKGATE_CTRL	Reserved					WBCP_VPW_THRESH		OSC_XTALOK_EN	OSC_XTALOK
W																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OSC_I		Reserved	STOP_MODE_CONFIG	Reserved				REFTOP_VBGUP	REFTOP_VBGADJ			REFTOP_SELFBIASOFF	Reserved		REFTOP_PWD
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XTALOSC24M_MISC0 field descriptions

Field	Description
31–29 -	This field is reserved.
28–26 CLKGATE_ DELAY	This field specifies the delay between powering up the XTAL 24MHz clock and releasing the clock to the digital logic inside the analog block. **NOTE:** Do not change the field during a low power event. This is not a field that the user would normally need to modify. 000 0.5ms 001 1.0ms 010 2.0ms 011 3.0ms 100 4.0ms 101 5.0ms 110 6.0ms 111 7.0ms
25 CLKGATE_CTRL	This bit allows disabling the clock gate (always ungated) for the xtal 24MHz clock that clocks the digital logic in the analog block. **NOTE:** Do not change the field during a low power event. This is not a field that the user would normally need to modify. 0 **ALLOW_AUTO_GATE** — Allow the logic to automatically gate the clock when the XTAL is powered down. 1 **NO_AUTO_GATE** — Prevent the logic from ever gating off the clock.
24–20 -	This field is reserved. Always set to zero.
19–18 WBCP_VPW_ THRESH	This signal alters the voltage that the pwell is charged pumped to. **NOTE:** Not related to oscillator. 00 **NOMINAL_BIAS** — Nominal output pwell bias voltage. 01 **PLUS_25MV** — Increase pwell output voltage by 25mV. 10 **MINUS_25MV** — Decrease pwell output pwell voltage by 25mV. 11 **MINUS_50MV** — Decrease pwell output pwell voltage by 50mV.
17 OSC_XTALOK_ EN	This bit enables the detector that signals when the 24MHz crystal oscillator is stable.
16 OSC_XTALOK	Status bit that signals that the output of the 24-MHz crystal oscillator is stable. Generated from a timer and active detection of the actual frequency.
15–14 OSC_I	This field determines the bias current in the 24MHz oscillator. The aim is to start up with the highest bias current, which can be decreased after startup if it is determined to be acceptable. 00 **NOMINAL** — Nominal 01 **MINUS_12_5_PERCENT** — Decrease current by 12.5% 10 **MINUS_25_PERCENT** — Decrease current by 25.0% 11 **MINUS_37_5_PERCENT** — Decrease current by 37.5%

Table continues on the next page...

XTALOSC24M_MISC0 field descriptions (continued)

Field	Description
13 Reserved	This field is reserved. Reserved
12 STOP_MODE_ CONFIG	Configure the analog behavior in stop mode. **NOTE:** Not related to oscillator. 0x0 **DEEP** — Deep Stop Mode - 0x0 All analog except RTC powered down on Stop mode assertion 0x1 **LIGHT** — Light Stop Mode - 0x1 All the analog domain except the LDO_1P1, LDO_2P5, and PLL3 is powered down on STOP mode assertion. If required the CCM can be configured not to power down the oscillator (XTALOSC). PLL3 can be disabled with register settings if desired.
11–8 -	This field is reserved. Reserved
7 REFTOP_ VBGUP	Status bit that signals the analog bandgap voltage is up and stable. 1 - Stable. **NOTE:** Not related to oscillator.
6–4 REFTOP_ VBGADJ	**NOTE:** Not related to oscillator. 000 Nominal VBG 001 VBG+0.78% 010 VBG+1.56% 011 VBG+2.34% 100 VBG-0.78% 101 VBG-1.56% 110 VBG-2.34% 111 VBG-3.12%
3 REFTOP_ SELFBIASOFF	Control bit to disable the self-bias circuit in the analog bandgap. The self-bias circuit is used by the bandgap during startup. This bit should be set after the bandgap has stabilized and is necessary for best noise performance of analog blocks using the outputs of the bandgap. **NOTE:** Value should be returned to zero before removing vddhigh_in or asserting bit 0 of this register (REFTOP_PWD) to assure proper restart of the circuit. **NOTE:** Not related to oscillator. 0 Uses coarse bias currents for startup 1 Uses bandgap-based bias currents for best performance.
2–1 -	This field is reserved.
0 REFTOP_PWD	Control bit to power-down the analog bandgap reference circuitry. **NOTE:** A note of caution, the bandgap is necessary for correct operation of most of the LDO, pll, and other analog functions on the die. **NOTE:** Not related to oscillator.

Appendix A
SDMA Scripts

A.1 Introduction

This appendix provides descriptions of scripts that may be used to perform data transfers using the Smart DMA (SDMA) block of the SoC.

The SDMA block supports data transfer from core memory space to core memory, from core memory space to peripherals, and vice versa.

A.1.1 SDMA Scripts Overview

Table A-1 gives an overview of the SDMA scripts.

Table A-1. SDMA Scripts Overview

Data Transfer	Script to Use	Use Case	Parameters Through Context	Parameters Through Buffer Descriptor
Memory	ap_2_ap	Memory Copy	None	Word length (2'b01 or 2'b10 or 2'b11 or 2'b00), Counter, first address is memory source address, second address is memory destination address
Shared UART	uartsh_2_mcu	Uart Rx	UART Rx FIFO address (r6) Event_mask (r1) Event2_mask (r0) Watermark level (r7)	Periph size (2'b01), Counter, Memory destination address, Second address not used
UART	uart_2_mcu	Uart Rx	Uart Rx FIFO address (r6) Event_mask (r1) Event2_mask (r0) Watermark level (r7)	Periph size (2'b01), Counter, Memory source address, Second address not used
SPDIF	mcu_2_spdif	Spdif Playback	SPDIF Tx FIFO address (r6) Event_mask (r1) Event2_mask (r0) Watermark level (r7)	Periph size (2'b10 or 2'b00), Counter, Memory source address, Second address not used

Table continues on the next page...

Table A-1. SDMA Scripts Overview (continued)

Data Transfer	Script to Use	Use Case	Parameters Through Context	Parameters Through Buffer Descriptor
	spdif_2_mcu	Spdif Record	SPDIF Rx FIFO address (r6) Event_mask (r1) Event2_mask (r0) Watermark level (r7)	Periph size (2'b10 or 2'b00), Counter, Memory source address, Second address not used
Shared Peripheral	mcu_2_shp	Peripheral Transmit	Tx fifo address (r6) Event_mask (r1) Event2_mask(r0) Watermark level (r7)	Periph size (2'b01 or 2'b10 or 2'b11 or 2'b00), Counter, Memory source address, Second address not used
	shp_2_mcu	Peripheral Receive	RX fifo address (r6) Event_mask (r1) Event2_mask(r0) Watermark level (r7)	Periph size (2'b01 or 2'b10 or 2'b11 or 2'b00), Counter, Memory destination address, Second address not used
Peripheral	mcu_2_app	Peripheral Transmit	Tx fifo address (r6) Event_mask (r1) Event2_mask(r0) Watermark level (r7)	Periph size (2'b01 or 2'b10 or 2'b11 or 2'b00), Counter, Memory source address, Second address not used
	app_2_mcu	Peripheral Receive	RX fifo address (r6) Event_mask (r1) Event2_mask(r0) Watermark level (r7)	Periph size (2'b01 or 2'b10 or 2'b11 or 2'b00), Counter, Memory destination address, Second address not used
SSI	mcu_2_ssiapp	Audio Playback	Tx FIFO 0 address (r6) Event_mask (r1) Event2_mask (r0) Watermark level (r7)	Periph size (2'b01 or 2'b10 or 2'b11 or 2'b00), Counter, Memory source address, Second address not used
	ssiapp_2_mcu	Audio Record	Rx FIFO 0 address (r6) Event_mask (r1) Event2_mask (r0) Watermark level (r7)	Periph size (2'b01 or 2'b10 or 2'b11 or 2'b00), Counter, Memory destination address, Second address not used
Shared SSI	mcu_2_ssish	Audio Playback	Tx FIFO 0 address (r6) Event_mask (r1) Event2_mask (r0) Watermark level (r7)	Periph size (2'b01 or 2'b10 or 2'b11 or 2'b00), Counter, Memory source address, Second address not used
	ssish_2_mcu	Audio Record	Rx FIFO 0 address (r6) Event_mask (r1) Event2_mask (r0) Watermark level (r7)	Periph size (2'b01 or 2'b10 or 2'b11 or 2'b00), Counter, Mem destination address, Second address not used
Peripheral	p_2_p	Peripheral Transfer	Destination address (r6) Event_mask (r1) Event2_mask (r0) Source address (r2) Information (r7)	Counter, First address and Second address not used
HDMI	hdmi_dma	HDMI playback	Buffer chain length (r4) Pointer address (r6)	None

In Table A-1, the data transfer column lists the possible types of DMA channels that involve the SDMA and a specific script is attached to each channel. It must be noted that some scripts cover several DMA channels. The scripts deal with the channels that have generic peripherals where only the watermark level is needed (no aging timer, no end_of_packet feature, and so on). The location refers to whether a script resides in ROM or in RAM and the size of the script is given in the instructions. The parameters columns are explained in the next section.

The following terms are used in Table A-1 which lists all the supported data transfers.

EMI or External Memory:

The external memories are connected to the External Memory Interface. Thus, a data transfer to EMI means a data transfer to any of the external memories (NAND Flash, NOR Flash, PSRAM, SDRAM, and so on).

Host mem or ARM internal memory:

The memory space accessible through the MAX of the ARM platform. It does not correspond to the peripherals, although they are accessible through the MAX as well. A data transfer from Host mem means data is read from the internal memory of the ARM or from the external memory. It is possible to point to an external memory through the MAX because it is also connected to EMI module. The next diagram replaces the SDMA and its DMA port in the hardware architecture. The Host mem is accessible through the Per DMA port of the SDMA, the EMI is accessible through the Burst DMA and the DSP mem through the DSP port.

Shared Peripheral:

A same peripheral can be connected to the shared peripheral bus (output of SPBA module) and to the ARM platform. This is the case for some UART, CSPI, and SSI. Therefore, "shared UART" indicates that the UART is connected to the shared peripheral, whereas "UART" means the UART is connected to the ARM platform.

Figure A-1 gives an overview of the SDMA in its hardware environment.

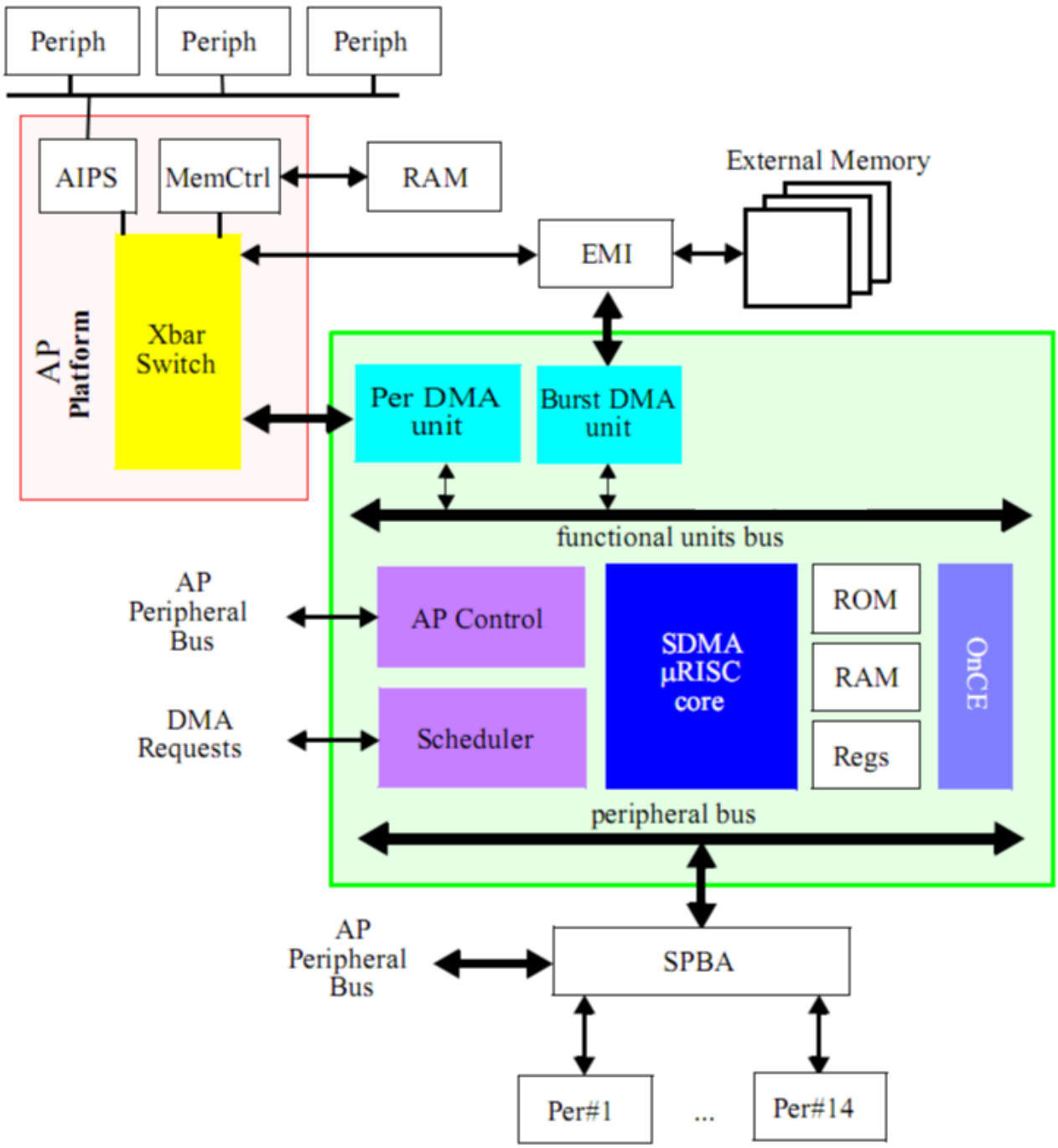


Figure A-1. SDMA Connections

A.2 Scripts Parameters: Definition

A.2.1 Parameters Definition

A.2.1.1 Parameters Required by Data Transfer Script Involving a Peripheral

The scripts that have a peripheral as a player in the transfer of data need to have the following input parameters:

A.2.1.1.1 Watermark Level (WML)

It is the upper or the lower threshold of the receive or transmit FIFO of the peripheral that triggers a DMA request to the SDMA.

The WML determines the data transfer loop size, meaning the number of bytes that will be read/write from/to the receive/transmit FIFO. This parameter must be a multiple of the peripheral FIFO data size. As an example, if UART1 is the data transfer destination, data must be written to the Tx FIFO of the peripheral. For instance, the watermark level can be set to 8 bytes (the UART is a 1 byte FIFO data size), meaning the UART_TX DMA request will be sent to the SDMA each time there are more than 8 free locations in the FIFO; therefore SDMA loop size will be set to 8 and 8 bytes will be written to the UART_TX FIFO. In fact, the loop size is the minimum between the WML and the count field of the buffer descriptor attached to the channel. This concept is explained later on in the chapter. Similarly, if the UART is the source of a data transfer and if the WML is set to 16, each time the DMA request is received by the SDMA, which triggers the associated channel, the data transfer loop size is set to 16. This means 16 bytes will be read from the UART_RX FIFO. The watermark level is a "long" (32-bit data) programmed by the ARM and is not supposed to evaluate a lot during application. The WML is always given in bytes.

NOTE

For the transmit FIFO, the WML = FIFO_SIZE - FIFOTX_THRESHOLD For the receive FIFO, the WML = FIFORX_THRESHOLD The FIFOTX_THRESHOLD/ FIFORX_THRESHOLD are the values of the peripheral transmit and receive FIFO level at which a DMA request is triggered. For instance, if TX FIFO is 32 bytes depth and if the DMA request is sent when the number of bytes present in the FIFO is below a threshold of 10, the WML will be 22. It means that when the SDMA will receive the DMA request from the

peripheral, it will be possible to store 22 bytes in the TX FIFO.
For the receive FIFO, the WML equals the threshold.

A.2.1.1.2 Event_mask and Event2_mask

As previously said, when the WML threshold is over or under passed, a DMA request is sent to the scheduler part of the SDMA.

The SDMA scheduler has 48 input pins, called "events", which are connected to the different DMA request.

The mapping between the DMA request name and the event number is SoC dependent and defined in the "Interrupts and SDMA Events" chapter. For instance, in SoC A, the UART_RX DMA request is connected to events pin 5; whereas for SoC B, it may be connected to events pin 15. The script reads the EVENTS(32 events requests) and EVENTS2(remaining 16 events requests) register to check if the received DMA request is compliant with the expected one and it guaranteed that a second DMA request sent during the data transfer is not lost.

For instance, while SDMA is reading the UART_RX FIFO (again, the number of data to be read is given by WML value), the peripheral may receive new data from its input ports. Therefore after the first data transfer, the number of data available in the UART received FIFO can be still higher than the WML threshold, so a second DMA request may be sent.

The event_mask value is generally a 1-bit high value, which means that if the script attached to the channel must be triggered by DMA request number I, event_mask[I] must be set to 1, if the script attached to the channel must be triggered by DMA request number 32+I, event2_mask[I] must be set to 1. Some scripts (like ata_2_mcu) may be triggered by several DMA requests; in that case several bits of the event_mask must be asserted. The event_mask and event2_mask are long (32-bit) data.

A.2.1.1.3 Peripheral Address

The scripts of the SDMA scripts library are SoC independent but as their goal is to address peripheral, they required the base address or the FIFO address of the peripheral.

Passing FIFO or peripheral base address depends on the scripts.

A.2.1.1.4 Data Length

The scripts whose name contains the key word "app" or "shp" are generic: they are used to transfer data from/to a 8, 16, 24 or 32-bit peripheral data size.

Some jumps to some routines of these scripts depend on this peripheral size. This parameter is required as input of these scripts. This parameter is passed through the command field of the buffer descriptor and is coded on bits 25 and 24 of the mode:

00: 32-bit data transfer.

01: 8-bit data transfer.

10: 16-bit datatransfer.

11: 24-bit data transfer.

A.3 Scripts

The following scripts are all based on buffer descriptor mechanism.

A.3.1 SDMA ROM Scripts

The ROM holds the generic memory to memory scripts (ap_2_ap, ap_2_bp, bp_2_ap, bp_2_bp) and the peripheral most useful scripts (app_2_mcu, mcu_2_app, uart_2_mcu, uartsh_2_mcu, mcu_2_shp, shp_2_mcu).

In these memory to memory scripts, the number of bytes to transfer from the source memory to the destination is divided by 4 to get the number of 32-bit words that can be transferred. Most of the time, the transfer will be a transfer of words. There will be a byte or half data transfer at the end of the transfer if the total number of bytes to transfer is not an multiple of 4.

A.3.1.1 ap_2_ap

This script is used to transfer data inside the application processor memory using BurstDMA.

Using this script there is no restriction on the number of bytes to transmit, the source address and destination address alignment, or on the source memory and destination memory type. But word alignment and source/destination memory type will impact performance. In case that the source and destination are in the same memory type, being word aligned allows the SDMA to use the copy capability of the DMA units. When it is not the case the SRAM of the SDMA is used as temporary buffer.

A.3.1.1.1 Parameters Transmitted Through the Context ap_2_ap

A.3.1.1.2 Parameters Transmitted Through the Buffer Descriptor ap_2_ap

The number of bytes to transmit is stored in the first Buffer descriptor word.

The source address is stored in the second Buffer descriptor word (address field).

The destination address is stored in the Extended Buffer address

A.3.1.1.3 Use of the General Register During the Execution ap_2_ap

- r0: nb of bytes to transfer, counter during loop
- r2: channel cb address, scratch
- r3: current buffer descriptor pointer, AP destination address, scratch
- r4: mode
- r5: source address loaded by getbd,
- r6: destination address loaded by getcb, scratch
- r7: pointer in the scratch buffer

A.3.1.1.4 Overview of Script Functionality ap_2_ap

The parameters of the transfer (number of bytes to transfer, source address and destination address) are retrieved from the buffer descriptor.

Depending on the address alignment and the DMA units involved, if possible, the transfer is performed using the copy capability of the DMA unit, or using the SDMA SRAM as a temporary buffer.

When the transfer is finished, this algorithm restarts (a new buffer descriptor is read).

The count in the BD mode word after the script is over is meaningful only if error was reported, else it has the same value as was fed as input to the script.

A.3.1.2 uartsh_2_mcu

This script is used to transfer data from the shared UART to the External memory. The UART is connected to the ARM platform.

The UART is an 8-bit peripheral, but when reading the FIFO, 16 bits are retrieved. In fact, the 8 MSB are the status bytes for the current data. At each access the value of this byte is checked to verify that a valid data was retrieved. If an error is detected the transfer is stopped and the information is sent back to the Host through the buffer descriptor with the byte count field updated.

A.3.1.2.1 Parameters Transmitted Through the Context uartsh_2_mcu

r0: mask to check events2 - If script is triggered by event 32+I, r0[I] must be set to 1. (Project dependent)

r1: mask to check event - If script is triggered by event I, r1[I] must be set to 1. (Project dependent)

r6: address of the peripheral Rx FIFO (project dependent)

r7: Watermark level - Used to determine the maximum of data that can be sent to the peripheral each time the channel is started.

A.3.1.2.2 Parameters Transmitted Through the Buffer Descriptor uartsh_2_mcu

The number of bytes to transmit is stored in the first Buffer descriptor word.

The destination address is stored in the second Buffer descriptor word (address field).

The Extended Buffer address is not used.

A.3.1.2.3 Use of the General Register During the Execution uartsh_2_mcu

r0 = mask to check events2

r1 = mask to check events

r2 = AP M3 destination address

r3 = Ram channel context address

r4 = Bd mode

r5 = Bd mode Count

r6 = SDMA memory mapped regs base address

r7 = Watermark

A.3.1.2.4 Overview of Script Functionality uartsh_2_mcu

When the UART sends a DMA request, it can be for three different reasons.

When the RX threshold is over passed. It deals with the most frequent event.

When the AGEING timer has reached its final value, meaning there is less data than the RX threshold in the RX FIFO, and they have been present for too long. It is called the aging DMA request.

When IDLE timer reached its final value; meaning the RX FIFO has been empty for too long. It is called the iddle dma request. The IDLE timer is not used, so only normal and AGEING DMA requests are supported by the script.

The first time the script is executed, a buffer descriptor is opened, the buffer destination address and its size are retrieved from the BD.

Then the script checks if the RRDY bit of USR1 is set. If yes, it means that the Watermark level has been reached, there are "WML" bytes in RX FIFO. If the count field of the opened BD is higher than WML-1, the SDMA script will perform WML-1 read accesses to the RX FIFO and WML write access to the destination buffer (a read access then a write access is call a data transfer loop). If the count field is lesser than the WML-1 value, the count field will be the data transfer loop size. So the data transfer loop size equals min(WML-1,BD count)

So there will be always one data remaining in the UART RX FIFO after every data transfer loop and the ageing timer will always trigger a DMA request. It means that the channel on which the script is run will be always closed on an AGEING DMA request.

Now assume the script is triggered by a DMA request, but RRDY is not set. It means that it deals with an AGEING DMA request; there is at least one data in the RX FIFO. The data is read and written into the destination buffer, and then the RDR is checked again. If it is set and if the destination data buffer is not full (BD count is not 0), a new data is read and so on until RDR is 0. When RDR is 0, the ageing timer interrupt flag is disabled by setting the AGTIM bit, the count field of BD is updated and the current BD is closed. Then the script tries to open the next BD if the Continuous bit of the current BD was set, if there is no more BD to open, the channel is stopped.

When reading a data from the UART RX FIFO, the MSB are check to verify that a valid data has been retrieved. If an error is detected the transfer is stopped, the error bit is set in the BD, the count is updated, the BD is closed and an interrupt is sent to the ARM.

When the buffer is full, this algorithm restarts (a new buffer descriptor is read, if it exists).

The next diagram illustrates the script algorithm, it is quite complex but things become more obvious when we notice that there is a loop for the normal DMA request (in sky-blue) and on when the AGEING DMA request was detected (in yellow).

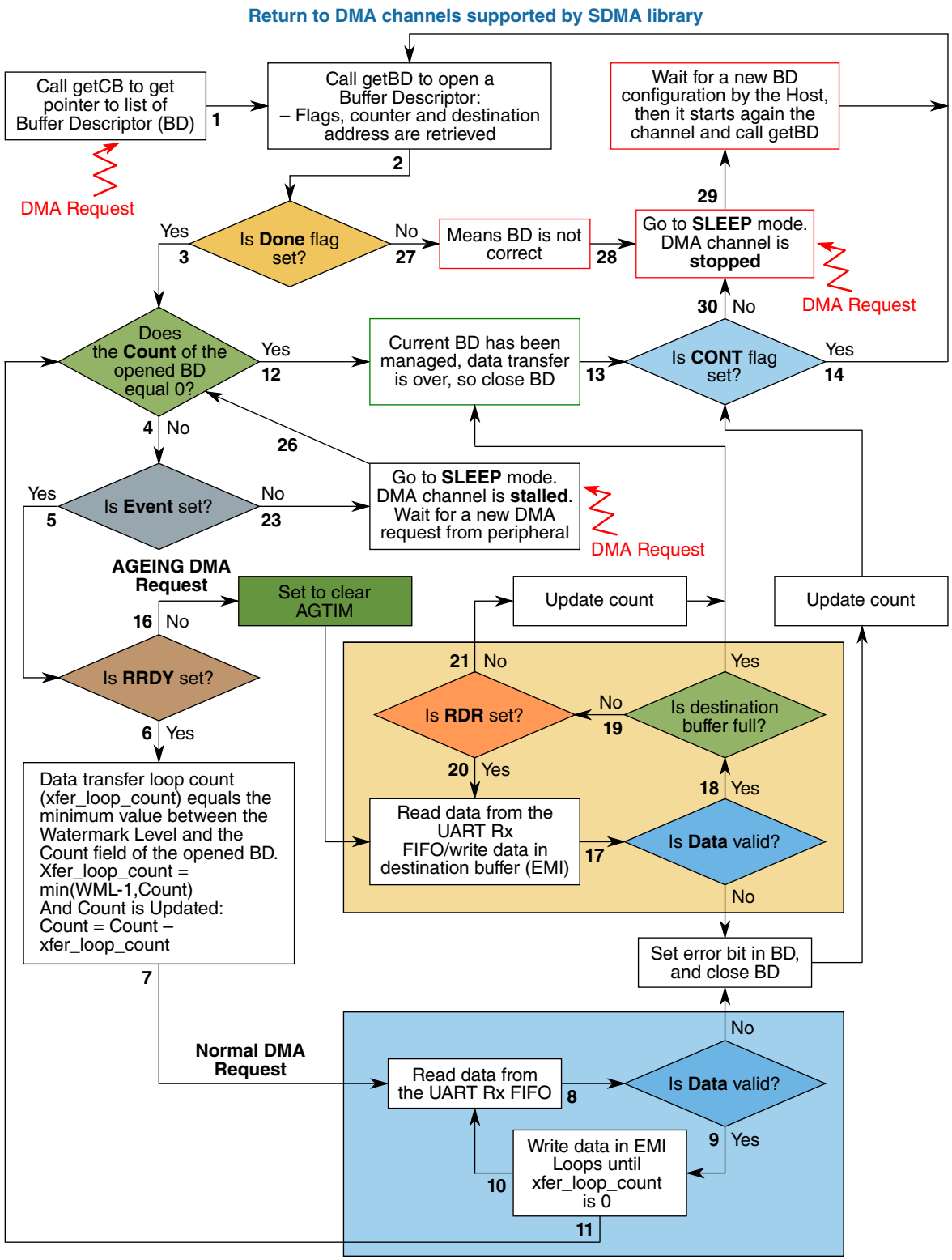


Figure A-2. Script Algorithm

A.3.1.3 shp_2_mcu

This generic script is used to transfer data from a 8, 16, 24 or 32-bit peripheral connected to the shared peripheral bus accessed through the SPBA to memories accessed by the BurstDMA (External memories).

It can be used for SSI (8, 16, 24 or 32-bit data size), for CSPI (32-bit data size) or for SDHC(MMC)/SIM (16-bit data size), these peripherals being connected to the shared peripheral bus.

A.3.1.3.1 Parameters Transmitted Through the Context shp_2_mcu

r0: mask to check events2 - If script is triggered by event 32+I, r0[I] must be set to 1. (Project dependent)

r1: mask to check event - If script is triggered by event I, r1[I] must be set to 1. (Project dependent)

r6: address of the peripheral Rx FIFO (project dependent)

r7: Watermark level - Used to determine the maximum of data that can be sent to the peripheral each time the channel is started.

A.3.1.3.2 Parameters Transmitted Through the Buffer Descriptor shp_2_mcu

The peripheral size/data length is set in the command field of the first Buffer descriptor word, especially bits 24 and 25.

The number of bytes to transmit is stored in the first Buffer descriptor word (count field)

The destination address in the external memory is stored in the second Buffer descriptor word (address field).

The Extended Buffer address is not used.

A.3.1.3.3 Use of the General Register During the Execution shp_2_mcu

r0 = mask to check events2

r1 = mask to check events

r2 = BD destination address

r3 = Ram channel context address

r4 = Bd mode

r5 = Bd mode Count

r6 = SDMA memory mapped regs base address

r7 = Watermark

A.3.1.3.4 Overview of Script Functionality shp_2_mcu

The parameters of the transfer (peripheral size, number of bytes to transfer and source address) are retrieved from the buffer descriptor.

When the peripheral sends an event, a data transfer loop is executed. This event is set when there is at least "watermark level" data in the FIFO.

During this data transfer loop, the number of bytes that are transferred from the RX FIFO of the peripheral to the EMI is equal to the minimum value between the "watermark level" and the number of data that have not been transferred yet. For each data transfer (one read, one write), the size of the memory write access is fixed by the parameter peripheral size.

The necessary number of transfer loops are executed in order to fill the buffer, but they are executed only if the event is set. While the event is not set, the channel is not executable.

When the buffer is filled, this algorithm restarts (a new buffer descriptor is read).

A.3.1.4 mcu_2_shp

This generic script is used to transfer data from memories accessed by the BurstDMA (External memories) to a 8, 16, 24 or 32-bit peripheral connected to the shared peripheral bus accessed through the SPBA

. It can be used for SSI (8, 16, 24 or 32-bit data size), for CSPI (32-bit data size), for SDHC(MMC)/SIM (16-bit data size) or for UART3,4 (8-bit data size), these peripherals being connected to the shared peripheral bus.

A.3.1.4.1 Parameters Transmitted Through the Context mcu_2_shp

r0: mask to check events2 - If script is triggered by event 32+I, r0[I] must be set to 1. (Project dependent)

r1: mask to check events - If script is triggered by event I, r1[I] must be set to 1. (Project dependent)

r6: address of the peripheral Tx FIFO (project dependent)

r7: Watermark level - Used to determine the maximum of data that can be retrieved from the peripheral each time the channel is started.

A.3.1.4.2 Parameters Transmitted Through the Buffer Descriptor mcu_2_shp

The peripheral size/data length is set in the command field of the first Buffer descriptor word, especially bits 24 and 25.

The number of bytes to transmit is stored in the first Buffer descriptor word (count field)

The source address in the external memory is stored in the second Buffer descriptor word (address field).

The Extended Buffer address is not used.

A.3.1.4.3 Use of the General Register During the Execution mcu_2_shp

r0 = mask to check events2

r1 = mask to check events

r2 = EVENTS / EVENTS2

r3 = Ram channel context address

r4 = Bd mode

r5 = Bd mode Count

r6 = SDMA memory mapped regs base address

r7 = Watermark

A.3.1.4.4 Overview of Script Functionality mcu_2_shp

The parameters of the transfer (peripheral size, number of bytes to transfer and source address) are retrieved from the buffer descriptor.

When the peripheral sends an event, a data transfer loop is executed. This event is set when there is at least "watermark level" empty rooms in the FIFO.

During this data transfer loop, the number of bytes that are transferred from the EMI to the TX FIFO of the peripheral is equal to the minimum value between the "watermark level" and the number of data that have not been transferred yet. For each data transfer (one read, one write), the size of the memory read access is fixed by the parameter peripheral size.

The necessary number of transfer loops are executed in order to empty the buffer, but they are executed only if the event is set. While the event is not set, the channel is not executable.

When the buffer is empty, this algorithm restarts (a new buffer descriptor is read).

A.3.1.5 uart_2_mcu

This script is similar to uartsh_2_mcu script. The only difference between these scripts is the way SDMA access UART RX FIFO.

In case of uartsh_2_mcu, the UART RX FIFO can directly be accessed by SDMA as it is on Shared Peripheral Bus. In case of uart_2_mcu, SDMA script uses one of its DMA (Peripheral DMA) to access UART RX FIFO.

A.3.1.6 app_2_mcu

This generic script is used to transfer data from a 8, 16, 24 or 32-bit peripheral connected to the AIPS accessed through the Periphera DMA of SDMA, to memories accessed by the BurstDMA (External memories).

It can be used for SSI (8, 16, 24 or 32-bit data size), for CSPI (32-bit data size) or for SDHC(MMC)/SIM (16-bit data size).

A.3.1.6.1 Parameters Transmitted Through the Context app_2_mcu

r0: mask to check events2 - If script is triggered by event 32+I, r0[I] must be set to 1. (Project dependent)

r1: mask to check event - If script is triggered by event I, r1[I] must be set to 1. (Project dependent)

r6: address of the peripheral Rx FIFO (project dependent)

r7: Watermark level - Used to determine the maximum of data that can be sent to the peripheral each time the channel is started.

A.3.1.6.2 Parameters Transmitted Through the Buffer Descriptor app_2_mcu

The Data Length peripheral size/data length is set in the command field of the first buffer descriptor word, specially bits 24 and 25.

The number of bytes to transmit is stored in the first buffer descriptor word (count field).

The destination address in the external memory is stored in the second buffer descriptor word (address field).

The Extended Buffer address is not used.

A.3.1.6.3 Use of the General Register During the Execution app_2_mcu

r0 = mask to check events2

r1 = mask to check events

r2 = AP M3 destination address

r3 = Ram channel context address

r4 = Bd mode

r5 = Bd mode Count

r6 = SDMA memory mapped regs base address

r7 = Watermark

A.3.1.6.4 Overview of Script Functionality app_2_mcu

The parameters of the transfer (peripheral size, number of bytes to transfer and source address) are retrieved from the buffer descriptor.

When the peripheral sends an event, a data transfer loop is executed. This event is set when there is at least "watermark level" data in the FIFO.

During this data transfer loop, the number of bytes that are transferred from the RX FIFO of the peripheral to the EMI is equal to the minimum value between the "watermark level" and the number of data that have not been transferred yet. For each data transfer (one read, one write), the size of the memory write access is fixed by the parameter peripheral size.

The necessary number of transfer loops are executed in order to fill the buffer, but they are executed only if the event is set. While the event is not set, the channel is not executable.

When the buffer is filled, this algorithm restarts (a new buffer descriptor is read).

A.3.1.7 mcu_2_app

This generic script is used to transfer data from memories accessed by the BurstDMA (External memories), to a 8, 16, 24 or 32-bit peripheral connected to the AIPS accessed through the Periphera DMA of SDMA.

It can be used for SSI (8, 16, 24 or 32-bit data size), for CSPI (32-bit data size) or for SDHC(MMC)/SIM (16-bit data size).

A.3.1.7.1 Parameters Transmitted Through the Context mcu_2_app

r0: mask to check events2 - If script is triggered by event 32+I, r0[I] must be set to 1. (Project dependent)

r1: mask to check event - If script is triggered by event I, r1[I] must be set to 1. (Project dependent)

r6: address of the peripheral Tx FIFO (project dependent)

r7: Watermark level - Used to determine the maximum of data that can be sent to the peripheral each time the channel is started.

A.3.1.7.2 Parameters Transmitted Through the Buffer Descriptor mcu_2_app

The peripheral size/data length is set in the command field of the first buffer descriptor word, specially bits 24 and 25.

The number of bytes to transmit is stored in the first buffer descriptor word (count field).

The source address in the external memory is stored in the second buffer descriptor word (address field).

The Extended Buffer address is not used.

A.3.1.7.3 Use of the General Register During the Execution mcu_2_app

r0 = mask to check events2

r1 = mask to check events

r2 = TX FIFO address

r3 = Ram channel context address

r4 = Bd mode

r5 = Bd mode Count

r6 = SDMA memory mapped regs base address

r7 = Watermark

A.3.1.7.4 Overview of Script Functionality mcu_2_app

The parameters of the transfer (peripheral size, number of bytes to transfer and source address) are retrieved from the buffer descriptor.

When the peripheral sends an event, a data transfer loop is executed. This event is set when there is at least "watermark level" empty room in the FIFO.

During this data transfer loop, the number of bytes that are transferred from the EMI to the TX FIFO of the peripheral is equal to the minimum value between the "watermark level" and the number of data that have not been transferred yet. For each data transfer (one read, one write), the size of the memory read access is fixed by the parameter peripheral size.

The necessary number of transfer loops are executed in order to fill the buffer, but they are executed only if the event is set. While the event is not set, the channel is not executable.

When the buffer is filled, this algorithm restarts (a new buffer descriptor is read).

A.3.1.8 mcu_2_spdif

This script performs a data move from AP buffer to 24 bits FIFO of the SPDIF.

This peripheral is on the peripheral bus, the SDMA accesses it through the SPBA. The MCU buffer is accessed through the burst DMA.

A.3.1.8.1 Parameters Transmitted Through the Context mcu_2_spdif

r0: mask to check events2 - If script is triggered by event 32+I, r0[I] must be set to 1. (Project dependent)

r1: mask to check events - If script is triggered by event I, r1[I] must be set to 1. (Project dependent)

r6: Address of SPDIF STL register

r7: Watermark level - Used to determine the maximum data that can be sent to the peripheral each time the channel is started.

A.3.1.8.2 Parameters Transmitted Through the Buffer Descriptor mcu_2_spdif

The data length (Either 32-bit or 16-bit) is set in the command field of the first Buffer descriptor word, especially bits 24, 25.

The number of bytes to transmit is stored in the first Buffer descriptor word (count field)

The source address in the external memory is stored in the second Buffer descriptor word (address field).

The Extended Buffer address is not used.

A.3.1.8.3 Use of the General Register During the Execution mcu_2_spdif

r0 = mask to check events2

r1 = mask to check events

r2 = EVENTS / EVENTS2

r3 = Ram channel context address

r4 = Bd mode

r5 = Bd mode Count

r6 = SDMA memory mapped regs base address

r7 = Watermark

A.3.1.8.4 Overview of Script Functionality mcu_2_spdif

The parameters of the transfer (data length, number of bytes to transfer and source address) are retrieved from the buffer descriptor.

When the spdif sends an event, a data transfer loop is executed. This event is set when there is at least "watermark level" of empty room in the FIFO (Sum of left and right FIFO's).

During this data transfer loop, the number of bytes that are transferred from the EMI to the TX FIFO of the spdif is equal to the minimum value between the "watermark level" and the number of data that have not been transferred yet. For each data transfer (one read, one write), the size of the memory read access is fixed by the parameter data length.

The necessary numbers of transfer loops are executed in order to empty the buffer, but they are executed only if the event is set. While the event is not set, the channel is not executable.

When the buffer is empty, this algorithm restarts (a new buffer descriptor is read).

A.3.1.9 spdif_2_mcu

This script performs a data move from 24 bits FIFO of the SPDIF to AP buffer.

This peripheral is on the peripheral bus, the SDMA accesses it through the SPBA. The MCU buffer is accessed through the burst DMA.

A.3.1.9.1 Parameters Transmitted Through the Context spdif_2_mcu

r0: mask to check events2 - If script is triggered by event 32+I, r0[I] must be set to 1. (Project dependent)

r1: mask to check events - If script is triggered by event I, r1[I] must be set to 1. (Project dependent)

r6: Address of SPDIF SRL register

r7: Watermark level - Used to determine the maximum data that can be sent to the peripheral each time the channel is started.

A.3.1.9.2 Parameters Transmitted Through the Buffer Descriptor spdif_2_mcu

The data length (Either 32 bit or 16 bit) is set in the command field of the first Buffer descriptor word, especially bits 24, 25.

The number of bytes to transmit is stored in the first Buffer descriptor word (count field)

The destination address in the external memory is stored in the second Buffer descriptor word (address field).

The Extended Buffer address is not used.

A.3.1.9.3 Use of the General Register During the Execution spdif_2_mcu

r0 = mask to check events2

r1 = mask to check events

r2 = EVENTS / EVENTS2

r3 = Ram channel context address

r4 = Bd mode

r5 = Bd mode Count

r6 = SDMA memory mapped regs base address

r7 = Watermark

A.3.1.9.4 Overview of Script Functionality spdif_2_mcu

The parameters of the transfer (data length, number of bytes to transfer and destination address) are retrieved from the buffer descriptor.

When the spdif sends an event, a data transfer loop is executed. This event is set when there is at least "watermark level" date in the FIFO (Sum of left and right FIFO's).

During this data transfer loop, the number of bytes that are transferred from the RX FIFO of the spdif to the EMI is equal to the minimum value between the "watermark level" and the number of data that have not been transferred yet. For each data transfer (one read, one write), the size of the memory write access is fixed by the parameter data length.

The necessary numbers of transfer loops are executed in order to empty the buffer, but they are executed only if the event is set. While the event is not set, the channel is not executable.

When the buffer is empty, this algorithm restarts (a new buffer descriptor is read).

A.3.2 SDMA RAM scripts

A.3.2.1 mcu_2_ssiapp

This script performs a data move from AP buffer to 8, 16, 24 or 32-bit FIFO of the SSI in the AP region with 2 FIFOs enabled.

The MCU buffer is accessed through the burst DMA, while SSI is accessed with Per DMA.

A.3.2.1.1 Parameters Transmitted Through the Context mcu_2_ssiapp

r0: mask to check events2 - If script is triggered by event 32+I, r0[I] must be set to 1. (Project dependent)

r1: mask to check events - If script is triggered by event I, r1[I] must be set to 1. (Project dependent)

r6: Address of SSI Transmit FIFO 0 register

r7: Watermark level - Used to determine the maximum data that can be sent to the peripheral each time the channel is started.

A.3.2.1.2 Parameters Transmitted Through the Buffer Descriptor mcu_2_ssiapp

The data length (8, 16, 24 or 32-bit) is set in the command field of the first buffer descriptor word, especially bits 24 and 25.

The number of bytes to transmit is stored in the first buffer descriptor word (count field).

The source address in the memory is stored in the second buffer descriptor word (address field).

The Extended Buffer address is not used.

A.3.2.1.3 Use of the General Register During the Execution mcu_2_ssiapp

r0 = mask to check events2

r1 = mask to check events

r2 = Tx FIFO address

r3 = Ram channel context address

r4 = Bd mode

r5 = Bd mode Count

r6 = SDMA memory mapped regs base address

r7 = Watermark

A.3.2.1.4 Overview of Script Functionality mcu_2_ssiapp

The parameters of the transfer (data length, number of bytes to transfer and source address) are retrieved from the buffer descriptor.

When the SSI sends an event, a data transfer loop is executed. This event is set when there is at least "watermark level" of empty room in the FIFO (Sum of left and right FIFO's).

During this data transfer loop, the number of bytes that are transferred from the EMI to the TX FIFO of the SSI is equal to the minimum value between the "watermark level" and the number of data that have not been transferred yet. For each data transfer (one read, one write), the size of the memory read access is fixed by the parameter data length.

The necessary numbers of transfer loops are executed in order to empty the buffer, but they are executed only if the event is set. While the event is not set, the channel is not executable.

When the buffer is empty, this algorithm restarts (a new buffer descriptor is read).

A.3.2.2 ssiapp_2_mcu

This script performs a data move from the 8, 16, 24 or 32-bit FIFO of the SSI to the AP buffer.

The SSI is in the AP region and 2 FIFO enabled. The MCU buffer is accessed through the burst DMA while the SSI is accessed through Per DMA.

A.3.2.2.1 Parameters Transmitted Through the Context ssiapp_2_mcu

r0: mask to check events2 - If script is triggered by event 32+I, r0[I] must be set to 1. (Project dependent)

r1: mask to check event - If script is triggered by event I, r1[I] must be set to 1. (Project dependent)

r6: address of the SSI RX FIFO 0

r7: Watermark level - Used to determine the maximum of data that can be sent to the peripheral each time the channel is started.

A.3.2.2.2 Parameters Transmitted Through the Buffer Descriptor ssiapp_2_mcu

The data length (8, 16, 24, 32 bits) is set in the command field of the first buffer descriptor word, especially bits 24 and 25.

The number of bytes to transmit is stored in the first buffer descriptor word (count field)

The destination address is stored in the second buffer descriptor word (address field).

The Extended Buffer address is not used

A.3.2.2.3 Use of the General Register During the Execution ssiapp_2_mcu

r0 = mask to check events2

r1 = mask to check events

r2 = AP M3 destination address

r3 = Ram channel context address

r4 = Bd mode

r5 = Bd mode Count

r6 = SDMA memory mapped regs base address

r7 = Watermark

A.3.2.2.4 Overview of Script Functionality ssiapp_2_mcu

When the SSI sends an event, a data transfer loop is executed. This event is set when there is at least "watermark level" of data in the FIFO(sum of left and right FIFOs).

During this data transfer loop, the number of bytes that are transferred from the RX FIFO of the SSI to the EMI is equal to the minimum value between the "watermark level" and the number of data that have not been transferred yet. For each data transfer (one read, one write), the size of the memory write access is fixed by the parameter data length.

The necessary numbers of transfer loops are executed in order to fill the buffer, but they are executed only if the event is set. While the event is not set, the channel is not executable.

When the buffer is full, this algorithm restarts (a new buffer descriptor is read).

A.3.2.3 mcu_2_ssish

This script is similar to mcu_2_ssiapp.

The only difference is the way SDMA core accesses the SSI TX FIFO. For script of mcu_2_ssiapp, SDMA core accesses the TX FIFO through Functional Unit Bus and regards it as the peripheral DMA, whereas for the mcu_2_ssish, the access is through the Shared Peripheral Bus and SSI can be directly accessed by SDMA.

A.3.2.4 ssish_2_mcu

This script is similar to ssiapp_2_mcu. The only difference is the way SDMA core accesses the SSI RX FIFO.

For script of ssiapp_2_mcu, SDMA core accesses the RX FIFO through Functional Unit Bus and regards it as the peripheral DMA, whereas for the ssish_2_mcu, the access is through the Shared Peripheral Bus and SSI can be directly accessed by SDMA.

A.3.2.5 p_2_p

This script performs a data move of 32 bits from the peripheral's FIFO connected either on SPBA or AIPS bus to another.

If destination FIFO belongs to ASRC then pad adding is performed after every N samples from the source device. If the source FIFO belongs to ASRC then after transmitting N samples a pad is swallowed. In case there is a transaction between ASRC(source) and SPDIF (destination) no pad adding or swallowing is required because SPDIF expects even number of samples.

This script presently does not deal with the transactions involving two devices connected to AIPS bus.

A.3.2.5.1 Parameters Transmitted Through the Context p_2_p

r0: LWML event mask

r1: HWML event mask

r2: source address

r6: destination address

r7: INFO. See details in the following table.

p_2_p (r7 INFO)

Bits	Name	Description
0-7	Lower WML	Watermark Level
8	PS	1 : Pad Swallowing 0 : No Pad swallowing
9	PA	1 : Pad Adding 0 : No Pad swallowing
10	SPDIF	If this bit is set both source and destination are on SPBA
11	Source Bit(SP)	1 : Source on SPBA 0 : Source on AIPS
12	Destination Bit (DP)	1 : Destination on SPBA 0 : Destination on AIPS
13-15	----------	MUST BE 0
16-23	Higher WML	HWML
24-27	N	Total number of samples after which Pad adding/Swallowing must be done. It must be odd.
28	Lower WML Event (LWE)	SDMA events reg to check for LWML event mask 0 : LWE in EVENTS register 1 : LWE in EVENTS2 register
29	Higher WML Event(HWE)	SDMA events reg to check for HWML event mask 0 : HWE in EVENTS register 1 : HWE in EVENTS2 register
30	----------	MUST BE 0
31	CONT	1 : Amount of samples to be transferred is unknown and script will keep on transferring samples as long as both events are detected and script must be manually stopped by the application 0 : The amount of samples to be is equal to the count field of mode word

A.3.2.5.2 Parameters Transmitted Through the Buffer Descriptor p_2_p

In the first buffer descriptor word, the mode and number of types to transfer is included.

The Buffer address and Extended Buffer address are not used.

A.3.2.5.3 Use of the General Register During the Execution p_2_p

The register usage for each case(spba_2_spba, asrc_2_spba…) are different, so don't present here.

A.3.2.5.4 Overview of Script Functionality p_2_p

The script get the number of data to transfer per event according to the bit 31(CONT) of INFO.

If CONT is 0, each transfer loop will transmit the number of data as in the mode count; Or else, it means the number of samples to be transferred is not known and the transfer must take place as long as both events are detected and the script must be manually stopped by the application in case the transfer needs to be stopped.

When PS or PA is set, then after each N words transfer, a pad swallowing or pad adding will be done. This is to handle the ASRC involved transfer.

The transfer is executed only when the event is set. After each transfer for the event, the script will try to obtain a new BD.

A.3.2.6 hdmi_dma

This script sets up the HDMI DMA start and end address from the buffer chain, and launches an HDMI DMA transfer on the HDMI DMA done event.

A.3.2.6.1 Parameters Transmitted Through the Context hdmi_dma

r4: Number of DMA buffers in loop

r6: Address of data structure setup by software. Detailed data structure below:

Table A-2. SDMA Scripts Overview

Name	Offset	Width (bytes)	Description
CONTROL	0	4	Address of HDMI DMA control register
STATUS	4	4	Address of HDMI DMA status register
START	8	4	Address of HDMI DMA start address register
START1	12	4	Start address of buffer 1 for HDMI DMA transfer
END1	16	4	End address of buffer 1 for HDMI DMA transfer
START2	20	4	Start address of buffer 1 for HDMI DMA transfer. Available only when value of r4>1
END2	24	4	End address of buffer 1 for HDMI DMA transfer. Available only when value of r4>1

Table continues on the next page...

Table A-2. SDMA Scripts Overview (continued)

Name	Offset	Width (bytes)	Description
STARTn	4+8*n	4	Start address of buffer 1 for HDMI DMA transfer. n is the value of r4.
ENDn	8+8*n	4	End address of buffer 1 for HDMI DMA transfer. n is ghe value in r4.

A.3.2.6.2 Parameters Transmitted Through the Buffer Descriptor hdmi_dma

None

A.3.2.6.3 Use of the General Register During the Execution hdmi_dma

r0 = N/A

r1 = control register address

r2 = status register address

r3 = DMA start address register address

r4 = offset of DMA buffer chain end

r5 = offset of current DMA buffer

r6 = pointer of data structure that described above

r7 = N/A

A.3.2.6.4 Overview of Script Functionality hdmi_dma

The parameters of the data structure used in the script are retrieved from the channel context. No parameter is retrieved through the buffer descriptor as no real SDMA transfer performed.

When the HDMI DMA done request is captured, the script will clear the DMA done bit in the status register and set up the DMA start and end address to the next buffer (rewind if it reaches the end of the buffer chain). Next it will restart the HDMI DMA transfer, then yield and wait for the next DMA done request.

When an error is captured, it will stop the channel.

Appendix B
i.MX 6Dual/6Quad Revision History

B.1 Substantive changes from revision 1 to revision 2

Substantive changes from revision 1 to revision 2 are as follows:

B.1.1 Reference Manual Revision History

No substantive changes

B.1.2 Architecture Overview Revision History

No substantive changes

B.1.3 Memory Maps Revision History

Reference	Description
ARM platform memory map	EIM naming updated as Memory and Register regions.

B.1.4 Interrupts Revision History

No substantive changes

B.1.5 External Signals Revision History

No substantive changes

B.1.6 Fusemap Revision History

Reference	Description
Fusemap	Updated column two of row "0x450[15:8] (BOOT_CFG2)" in SD/eSD Boot Fusemap from "Reserved" to contain a description.
Fusemap Description Table	Updated Speed Grading value from 850 MHz to 852 MHz.
Fusemap	Updates to BOOT_CFG4[7].
Fusemap Description Table	Added row for temperature grade.

B.1.7 External Memory Controllers Revision History

No substantive changes

B.1.8 System Debug Revision History

No substantive changes

B.1.9 System Boot Revision History

Reference	Description
IOMUX Configuration for SD/MMC	SD4 DAT[7:0] alt modes and RESET pin updated.
Write Data Command	EIM split regions updated with Memory and Register identifiers.
NAND eFUSE Configuration	Updated BOOT_CFG1[1:0] fuse definition to read "Row Address Cycles" vs. "Adress Cycles".
HAB API Vector Table Addresses	Added vector address table.

B.1.10 Multimedia Revision History

B.1.11 Power Management Revision History

Reference	Description
Examples of External Power Supply Interfacing in the i.MX 6Dual/6Quad based systems	Updated text in figures regarding connection to power supply.

B.1.12 System Security Revision History

No substantive changes

B.1.13 ARM A9 Revision History

No substantive changes

B.1.14 AIPSTZ Revision History

Reference	Description
-	Updated memory map.

B.1.15 APBH Revision History

No substantive changes

B.1.16 ASRC Revision History

No substantive changes

B.1.17 AUDMUX Revision History

Reference	Description
Normal Mode	Updated text to read, "PTCR2's RFSEL[3:0] and RCSEL[3:0] must be set to 011b while PTCR4's RFSDIR and RCLKDIR set to 0." Previously read, "...RCKDIR set to 1".

B.1.18 BCH Revision History

Reference	Description
BCH Memory Map/Register Definition	SET, CLEAR, TOGGLE (SCT) registers added

B.1.19 CCM Revision History

Reference	Description
CCM Memory Map/Register Definition	CBCDR and CBCMR reset values updated.
-	MMDC clock naming updated
Clock Root Generator	Clock root generator figures updated.
CCM Memory Map/Register Definition	CCM_CLPCR register updated.
External Signals	External Signal table updated.
CCM Clock Tree	Clock tree updated
Entering WAIT mode	Removed some internal information re: gating.
CCM Clock Tree	VIDEO_27M and UART clock roots updated.
System Clocks	Updated max freq of IPU1_HSP_CLK_ROOT and IPU2_HSP_CLK_ROOT to 264 from 266.
CCM Clock Tree	Clock tree updated
Clock Root Generator	Branch figures updated
CCM Memory Map/Register Definition	Minor updates to clock names. PLL3 inputs to clock root muxes updated to pll3_sw_clk.
Clock Switching Multiplexers	New topic added
-	References to pll2_sw_clk and related programming removed
System Clocks	System clocks, gating and override table updated

B.1.20 CSI2IPU Revision History

No substantive changes

B.1.21 DCIC Revision History

No substantive changes

B.1.22 ECSPI Revision History

No substantive changes

B.1.23 EIM Revision History

Reference	Description
Burst (Synchronous Mode) Write Memory Access Timing - BCD=1	Added new topic to timing diagram section.

B.1.24 ENET Revision History

Reference	Description
External Signals	Removed some text re: RGMII mode throughout table.

B.1.25 EPIT Revision History

No substantive changes

B.1.26 ESAI Revision History

Reference	Description
ESAI Memory Map/Register Definition	Updated THCKP bit field description in the ESAI_TCCR register.

B.1.27 FLEXCAN3 Revision History

No substantive changes

B.1.28 GPC Revision History

No substantive changes

B.1.29 GPIO Revision History

Reference	Description
GPIO pad structure	GPIO pad structure topics added.

B.1.30 GPMI Revision History

No substantive changes

B.1.31 GPT Revision History

No substantive changes

B.1.32 GPU2D Revision History

No substantive changes

B.1.33 GPU3D Revision History

No substantive changes

B.1.34 HDMI Revision History

Reference	Description
HDMI Memory Map/Register Definition	Updated HDMI_IH_MUTE register.
HDMI Memory Map/Register Definition	HDMI_FC_INVIDCONF register definition added

B.1.35 HDMI PHY Revision History

No substantive changes

B.1.36 I2C Revision History

No substantive changes

B.1.37 IOMUXC Revision History

Reference	Description
-	DRAM_RESET setting updated. Pull/Keep functionality disabled on DRAM pads with exception of DRAM_DATAn and DRAM_SDQSn.
-	Cross-reference links added to pad groups.
-	Pad control register DSE fields for GPIO pad types (EIM, DI, ENET, NAND, CSI, SD, KEY, DISP, GPIO) updated.
-	SPEED field values updated for GPIO Pad types. Note added to related SPEED and SRE fields.

B.1.38 IPU Revision History

Reference	Description
Camera Ports	Updated maximum speed for parallel interface to 240MHz.
Processing flows	VF3 removed from Camera Preview.

B.1.39 KPP Revision History

No substantive changes

B.1.40 LDB Revision History

No substantive changes

B.1.41 MIPI CSI Revision History

No substantive changes

B.1.42 MIPI DSI Revision History

No substantive changes

B.1.43 MIPI HSI Revision History

No substantive changes

B.1.44 MLB150 Revision History

No substantive changes

B.1.45 MMDC Revision History

Reference	Description
MMDC initialization	ZQ calibration added to initialization steps.
ZQ calibration	ZQ calibration description updated.
Overview	Clock speeds updated.
MMDC Memory Map/Register Definition	fast_clk references updated to MMDC_CH0_AXI clock (fast clock)
MMDC Profiling	Added information regarding busy cycles in MADPSR1 (Busy cycles count) bullet.
MMDC Memory Map/Register Definition	Updated MMDC_MDASP bitfield description.
MMDC Memory Map/Register Definition	Added note to WALAT bitfield description.
External Signals	External signal direction updated.

B.1.46 NIC Revision History

No substantive changes

B.1.47 OCOTP Revision History

No substantive changes

B.1.48 OCRAM Revision History

No substantive changes

B.1.49 PCIe Ctrl Revision History

No substantive changes

B.1.50 PCIe PHY Revision History

No substantive changes

B.1.51 PMU Revision History

Reference	Description
PMU	Updates to PMU_REG_Core.
Digital LDO Regulators	New note at end of section.

B.1.52 PWM Revision History

No substantive changes

B.1.53 ROMCP Revision History

Reference	Description
ROMCP Memory Map/Register Definition	ROMCP Data Register instance value order updated.

B.1.54 SATA Revision History

No substantive changes

B.1.55 SATA PHY Revision History

No substantive changes

B.1.56 SDMA Revision History

Reference	Description
Low Power Modes and User Control	Removed CRC column from power modes table.

B.1.57 SJC Revision History

No substantive changes

B.1.58 SNVS Revision History

No substantive changes

B.1.59 SPBA Revision History

No substantive changes

B.1.60 SPDIF Revision History

No substantive changes

B.1.61 SRC Revision History

No substantive changes

B.1.62 SSI Revision History

No substantive changes

B.1.63 TEMPMON Revision History

No substantive changes

B.1.64 TZASC Revision History

No substantive changes

B.1.65 UART Revision History

Reference	Description
External Signals	New figure for external signal path in DCE and DTE mode.

B.1.66 USB Revision History

No substantive changes

B.1.67 USB PHY Revision History

Reference	Description
USB_ANALOG	Update register bit description of USB_ANALOG_DIGPROG.

B.1.68 USDHC Revision History

No substantive changes

B.1.69 VDOA Revision History

No substantive changes

B.1.70 VPU Revision History

Reference	Description
Clocks	Update to the cclk row.

B.1.71 WDOG Revision History

Reference	Description
Watchdog mechanism and system integration	Added new topic

B.1.72 XTALOSC Revision History

Reference	Description
Overview	Update to block overview list.
Oscillator Configuration (32 kHz)	Updates to second paragraph.

B.1.73 SDMA Scripts Revision History

No substantive changes

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: IMX6DQRM
Rev. 2
06/2014